UNIVERSITÀ DEGLI STUDI
DI TRENTO

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE
ICT International Doctoral School

Efficient and Effective Solutions for Video Classification

Ionut Cosmin Duta

Advisor:

Prof. Nicu Sebe

University of Trento

Co-Advisor:

Prof. Bogdan Ionescu

University Politehnica of Bucharest

November 2017

Contents

PUBLICATIONS

This thesis consists of the following publications:

- Chapter 2:

  Uijlings, J.R.R.; Duta, I.C.; Sangineto, E. and Sebe, N. "Video classification with Densely extracted HOG/HOF/MBH features: an evaluation of the accuracy/computational efficiency trade-off". In International Journal of Multimedia Information Retrieval, 4(1):33-44, 2015.

  Idea previously appeared in:

  Uijlings, J.R.R.; Duta, I.C.; Rostamzadeh, N. and Sebe, N. "Real-time video classification using dense HOF/HOG". In International Conference on Multimedia Retrieval (ICMR), 2014.

- Chapter 3:

  Duta, I.C.; Uijlings, J.R.R.; Aizawa, K.; Hauptmann, A.G.; Ionescu, B. and Sebe, N. "Efficient Human Action Recognition using Histograms of Motion Gradients and VLAD with Descriptor Shape Information". In Multimedia Tools and Applications (MTAP), DOI: 10.1007/s11042-017-4795-6, 2017.

  Part of the idea previously appeared in:

  Duta, I.C.; Uijlings, J.R.R.; Nguyen, T. A.; Aizawa, K.; Hauptmann, A.G.; Ionescu, B. and Sebe, N. "Histograms of Motion Gradients for Real-time Video Classification" In International Workshop on Content-based Multimedia Indexing (CBMI), 2016.

- Chapter 4:

  Duta, I.C.; Ionescu, B.; Aizawa, K. and Sebe, N. "Spatio-Temporal Vector of Locally Max Pooled Features for Action Recognition in Videos". In Computer Vision and Pattern Recognition (CVPR), 2017.

The papers published during the course of the PhD but not included in this thesis are the following:

- Duta, I.C.; Ionescu, B.; Aizawa, K. and Sebe, N. "Simple, Efficient and Effective Encodings of Local Deep Features for Video Action Recognition". In International Conference on Multimedia Retrieval (ICMR), 2017.

- Duta, I.C.; Ionescu, B.; Aizawa, K. and Sebe, N. "Spatio-temporal VLAD Encoding for Human Action Recognition in Videos". In International Conference on Multimedia Modeling (MMM), 2017.

- Duta, I.C.; Nguyen, T.A.; Aizawa, K.; Ionescu, B. and Sebe, N. "Boosting VLAD with Double Assignment using Deep Features for Action Recognition in Videos". In International Conference on Pattern Recognition (ICPR), 2016.

- Mironica, I.; Duta, I.C.; Ionescu, B. and Sebe, N. "A modified vector of locally aggregated descriptors approach for fast video classification". In Multimedia Tools and Applications (MTAP), 75(15):9045-9072, August 2016.

- Mironica, I.; Duta, I.C.; Ionescu, B. and Sebe, N. "Beyond bag-of-words: Fast video classification with fisher kernel vector of locally aggregated descriptors". In International Conference on Multimedia and Expo (ICME), 2015.

Chapter 1

Introduction

Video understanding is one of the long-standing goals of the computer vision and multimedia communities. The ability to automatically understand video content opens the door to a huge pool of potential applications such as automatic video analysis, video indexing and retrieval, video surveillance, virtual reality, human-computer interaction, robot learning, etc. Video is one of the most notorious multimedia content for entertainment and communication. A statistical study[1] reveals that on YouTube there are currently updated more than 300 hours of video every minute, each day YouTube users watch more than a billion hours of video. This explosive growth in video content continues to have a fulminant increase, for instance Cisco forecast[2] mentioned that the IP video would account for 80% of all IP traffic by 2019. Given this fulminant growth in video content, the capability of computers to perform video classification becomes very challenging and crucial for various purposes such as search, recommendation, ranking, etc.

Computers are far behind humans in understanding video content. Besides the enormous amount of video content, it is very challenging to perform video classification due to many reasons such as large intra-class varia-

[1] https://www.youtube.com/yt/about/press/

[2] http://newsroom.cisco.com/press-release-content?articleId=1644203

tions, viewpoint changes, background clutter, high dimension of video data, low video resolution, camera motion, etc. All of these make video classification a very challenging and computationally demanding task, however, video classification has received a sustained attention from the research community due to the unlimited potential of real-life applications.

The aim of this PhD thesis is to make a step forward towards teaching computers to understand videos in a similar way as humans do. In this work we tackle the video classification and/or action recognition tasks. This thesis was completed in a period of transition, the research community moving from traditional approaches (such as hand-crafted descriptor extraction) to deep learning. Therefore, this thesis captures this transition period, however, unlike image classification, where the state-of-the-art results are dominated by deep learning approaches, for video classification the deep learning approaches are not so dominant. As a matter of fact, most of the current state-of-the-art results in video classification are based on a hybrid approach where the hand-crafted descriptors are combined with deep features to obtain the best performance. This is due to several factors, such as the fact that video is a more complex data as compared to an image, therefore, more difficult to model and also that the video datasets are not large enough to train deep models with effective results.

The pipeline for video classification, illustrated in Figure 1.1, can be broken down into three main steps: feature extraction, encoding and classification. While for the classification part, the existing techniques are more mature, for feature extraction and encoding there is still a significant room for improvement. In addition to these main steps, the framework contains some pre/post processing techniques, such as feature dimensionality reduction, feature decorrelation (for instance using Principal Component Analysis - PCA) and normalization, which can influence considerably the performance of the pipeline.

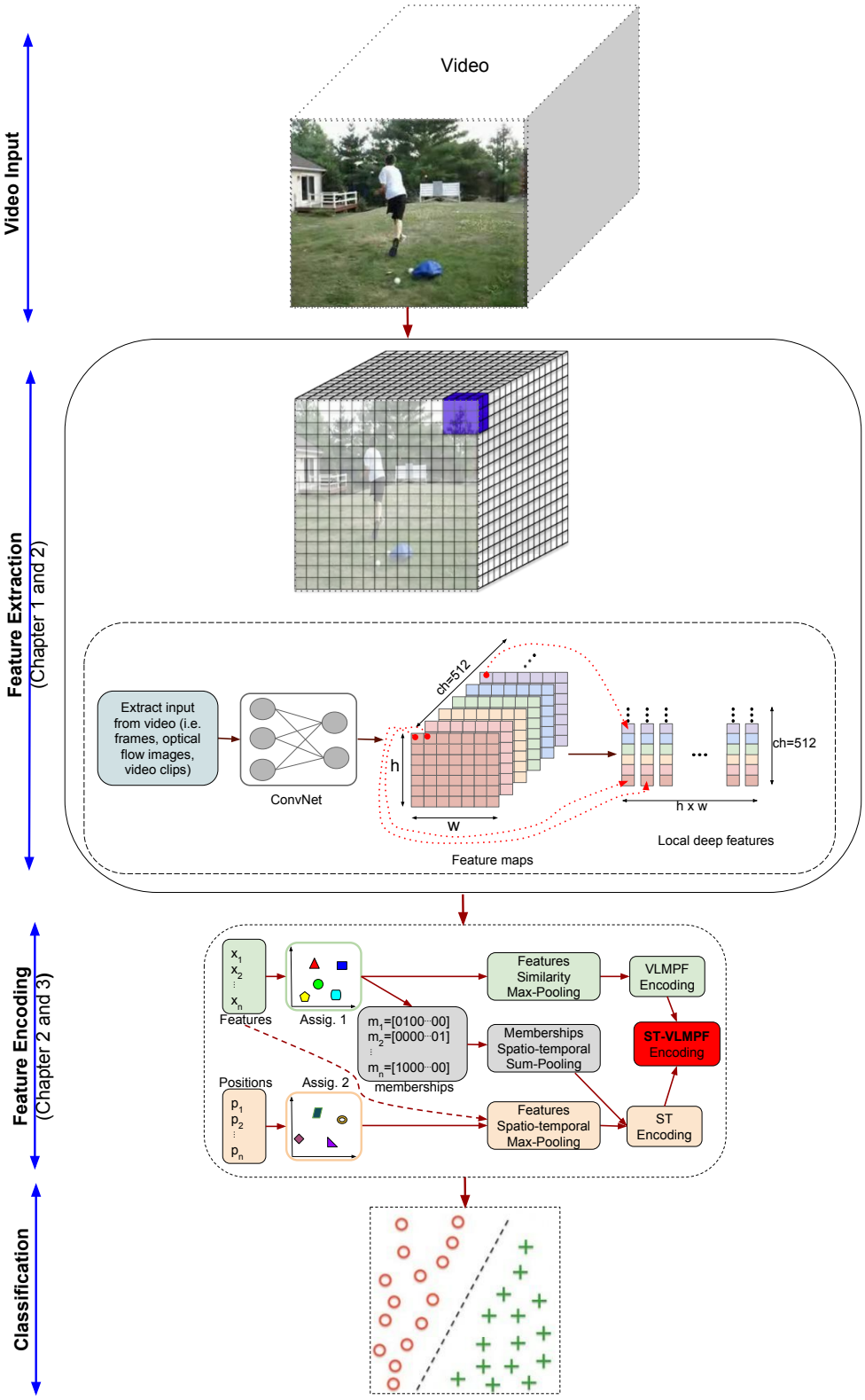

Figure 1.1: The framework for video classification.

This thesis is mainly based on three works. The first work focuses on descriptor extraction, which is an important step in the video classification pipeline. The second and the third works tackle the feature encoding part which is highly dependent on the descriptor extraction step. At the end, the feature encoding approach drastically influences the classifier performance.

One of the bottlenecks of the video classification pipeline is represented by the feature extraction step, where most of the approaches are extremely computationally demanding, what makes them not suitable for real-time applications. In this thesis, we tackle this issue, propose different speed-ups to improve the computational cost and introduce a new descriptor that can capture motion information from a video without the need of computing optical flow (which is very expensive to compute). Another important component for video classification is represented by the feature encoding step, which builds the final video representation that serves as input to a classifier. During the PhD, we proposed several improvements over the standard approaches for feature encoding. We also propose a new feature encoding approach for deep feature encoding. To summarize, the main contributions of this thesis are as follows[3]:

- We propose several speed-ups for descriptor extraction, providing a version for the standard video descriptors that can run in real-time. We also investigate the trade-off between accuracy and computational efficiency.

- We provide a new descriptor for extracting information from a video, which is very efficient to compute, being able to extract motion information without the need of extracting the optical flow.

- We investigate different improvements over the standard encoding approaches for boosting the performance of the video classification

[3]Source code is available at: `https://iduta.github.io/software.html`

pipeline.

- We propose a new feature encoding approach specifically designed for encoding local deep features, providing a more robust video representation.

The remainder of this thesis is organized as follows:

- In Chapter 2, we present our work on descriptor extraction for video classification, where we address one of its issues: high computational cost. Specifically, the contributions are: (1) We propose several speed-ups for densely sampled HOG (Histogram of Orientated Gradients), HOF (Histogram of Optical Flow) and MBH (Motion Boundary Histograms) descriptors and release Matlab code; (2) We investigate the trade-off between accuracy and computational efficiency of descriptors in terms of frame sampling rate and type of Optical Flow method; (3) We investigate the trade-off between accuracy and computational efficiency for computing the feature vocabulary, using and comparing most of the commonly adopted vector quantization techniques: k-means, hierarchical k-means, Random Forests, Fisher Vectors and Vector of Locally Aggregated Descriptors (VLAD).

- Chapter 3 continues with the work on descriptor extraction and also makes the transition from descriptor extraction to feature encoding. Specifically, the contributions are: (1) We introduce a new descriptor, which captures the motion information using a simple temporal derivation, without the need of using the costly optical flow. We make the code for descriptor extraction available. (2) We propose a new encoding method, which captures shape information within the encoding process, providing the best trade-off between accuracy and computational cost. We make the code for descriptor encoding available. (3) We adopt several speed-ups, such as fast aggregation of gradient

responses, reuse subregions of aggregated magnitude responses, and frame subsampling, which make the pipeline more efficient. (4) We propose an integration of our descriptor and encoding method in a specifically designed video classification framework which allows for real-time performance while maintaining the high accuracy of the results.

- Chapter 4 continues the work on feature encoding. Specifically, the contributions are: (1) Provide a new encoding approach specifically designed for working with deep features. We exploit the nature of deep features, with the goal of capturing the highest feature responses from the highest neuron activation of the network. (2) Efficiently incorporate the spatio-temporal information within the encoding method by taking into account the features position and specifically encode this aspect. Spatio-temporal information is crucially important when dealing with video classification. (3) Provide an action recognition scheme to work with deep features, which can be adopted to obtain impressive results with any already trained network, without the need for re-training or fine tuning on a particular dataset. Furthermore, our framework can easily combine different information extracted from different networks. In fact, our pipeline for action recognition provides a reliable representation outperforming the previous state-of-the-art approaches, while maintaining a low complexity.

- Chapter 5 concludes this thesis and presents the future research work.

Chapter 2

Video classification with densely extracted HOG/HOF/MBH features: an evaluation of the accuracy/computational efficiency trade-off[1]

A widely used framework in video classification is based on Bag-of-Words using local visual descriptors. Most commonly these are Histogram of Oriented Gradient (HOG), Histogram of Optical Flow (HOF) and Motion Boundary Histogram (MBH) descriptors. While such approach is very powerful for classification, it is also computationally expensive. This work addresses the problem of computational efficiency. Specifically: (1) We propose several speed-ups for densely sampled HOG, HOF and MBH descriptors and release Matlab code; (2) We investigate the trade-off between accuracy and computational efficiency of descriptors in terms of frame sampling rate and type of Optical Flow method; (3) We investigate the trade-off

[1]Uijlings, J.R.R.; Duta, I.C.; Sangineto, E. and Sebe, N. "Video classification with Densely extracted HOG/HOF/MBH features: an evaluation of the accuracy/computational efficiency trade-off". In International Journal of Multimedia Information Retrieval, 4(1):33-44, March 2015.

between accuracy and computational efficiency for computing the feature vocabulary, using and comparing most of the commonly adopted vector quantization techniques: k-means, hierarchical k-means, Random Forests, Fisher Vectors and VLAD.

2.1 Introduction

The Bag-of-Words method [14, 72] has been successfully adapted from the domain of still images to the domain of video by using local, visual, space-time descriptors (e.g. [46, 19, 41, 66, 67, 86]). Successful applications range from Human Action Recognition [46, 44, 63] to Event Detection [73] and Concept Classification [74, 73]. However, analyzing video is even more computationally expensive than analysing images. Hence, in order to deal with the enormous, growing amount of digitalized video it is important to have not only accurate, but also computationally efficient methods.

In this work we take a powerful, commonly used Bag-of-Words pipeline for video classification and investigate how we can make it more computationally efficient while sacrificing as little accuracy as possible. The general pipeline is visualized in Figure 2.1. In this pipeline we focus on densely sampled local visual descriptors only, since dense sampling has been found to be more accurate than keypoint-based sampling, both in images [37] and in video [90]. As type of local visual descriptors, we focus on the standard ones, which are based on local 3D volumes of Histograms of Oriented Gradients (HOG) [15], Histograms of Optical Flow (HOF) [16, 46] and Motion Boundary Histograms (MBH) [16]. For transforming the set of local descriptors extracted from a video into a fixed-length vector necessary for classification, we compare a variety of techniques: k-means, hierarchical k-means, Random Forests [7, 31], Fisher Vectors [61] and Vector of Locally Aggregated Descriptors (VLAD) [35]. Starting from this pipeline,

this evaluation chapter makes the following contributions:

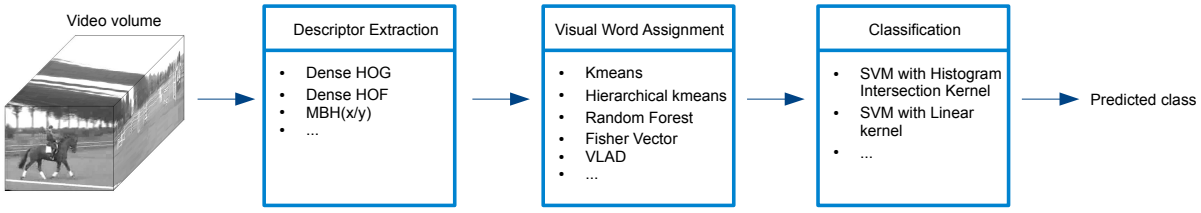

Figure 2.1: General framework for video classification using a Bag-of-Words pipeline. The methods evaluated in this work are instantiated in this diagram.

Fast Dense HOG/HOF/MBH. We exploit the nature of densely sampled descriptors in order to speed up their computation. HOG, HOF and MBH descriptors are created from subvolumes. These subvolumes can be shared by different descriptors similar to what was done in [83]. In this work we generalize their idea of reusing subregions to 3 dimensions. Matlab source code is available[2].

Evaluation of frame subsampling. Videos consist of many frames, making them computational expensive to analyze. However, subsequent frames also largely carry the same information. In this work we evaluate the trade-off between accuracy and computational efficiency when subsampling video frames.

Evaluation of Optical Flow. Calculating optical flow is generally expensive and takes up much of the total HOF and MBH descriptor extraction time. But for optical flow there is also a trade-off between computational efficiency and accuracy. Moreover, optical flow methods are generally tested against optical flow benchmarks such as [3, 10], but it is not immediately obvious that methods which perform well on these benchmarks would automatically also yield better HOF and MBH descriptors.

[2] `https://iduta.github.io/software.html`

Therefore in this work we evaluate optical flow methods directly in our task of interest: video classification. Specifically, we compare the optical flow methods of Lukas-Kanade [49], Horn-Schunk [32], Farnebäck [28], Brox 04 [8], and Brox 11 [9].

Evaluation of descriptor encoding. The classical way of transforming a set of local visual descriptors into a single fixed-length vector is by using a k-means visual vocabulary and assign local descriptors to the mean of the nearest cluster (e.g. [14]). However, both hierarchical k-means and Random Forests [53, 83] are viable fast alternatives. Furthermore, the Fisher Vector [61] significantly outperforms classical k-means representation in many tasks, whereas VLAD [35] can be considered a simplified non-probabilistic version of the Fisher Vector [64] and it is computationally more efficient. In this work we evaluate the accuracy/efficiency trade-off of all five methods above in the context of video classification.

2.2 Related work

The most used local spatio-temporal descriptors are modeled after SIFT [48]: each local video volume is divided into blocks, for each block one aggregates responses (either oriented gradients or optical flow), and the final descriptor is a concatenation of the aggregated responses of several adjacent blocks. Both Dalal et al. [16] and Laptev et al. [46] proposed to aggregate 2D Oriented Gradient Responses (HOG) and Optical Flow responses (HOF). Additionally, Dalal et al. [16] also proposed to calculate changes of optical flow, or Motion Boundary Histograms (MBH). Both Scovanner et al. [67] and Kläser et al. [41] proposed to measure oriented gradients also in the temporal dimension, resulting in 3-dimensional gradient responses. Everts et al. [26] extended [41] to include color channels. As Wang et al. [90] found little evidence that the 3D responses of [41] are better than HOG,

in this chapter we implemented and evaluated the descriptors which are most widely used: HOG, HOF and MBH.

Wang et al. [90] evaluated several interest point selection methods and several spatio-temporal descriptors. They found that dense sampling methods generally outperform interest points, especially on more difficult datasets. As this result was earlier found in image analysis [37, 65], this work focuses on dense sampling for videos. In [90] the evaluation was on accuracy only. In contrast, this work focuses on the trade-off between computational efficiency and accuracy.

Wang et al. [86] proposed to use dense trajectories. In their method, the local video volume moves spatially through time; it tries to stay on the same part of the object. Additionally, they use changes in optical flow rather than the optical flow itself. They show good improvements over normal HOG, HOF and MBH descriptors. Nevertheless, combining their dense trajectory descriptors with both normal HOG, HOF and MBH descriptors still gives significant improvements over dense trajectories alone [39, 86]. In this work we focus on HOG, HOF and MBH. Note that we evaluate the accuracy/efficiency trade-off for several optical flow methods which may be of interest also when using dense trajectories.

In [65], Sangineto proposes to use Integral Images [85] to efficiently compute densely extracted SURF features [5] in *still images*. The work of Uijlings et al. [83] proposes several methods to speed up the Bag-of-Words classification pipeline for *image* classification and provides a detailed evaluation on the trade-off between computational efficiency and classification accuracy. In this work we perform such evaluation on *video* classification. Inspired by [83] we propose accelerated densely extracted HOG, HOF and MBH descriptors and provide efficient Matlab implementations. Additionally, we evaluate various video-specific aspects such as frame sampling rate and the choice of optical flow method.

The Fisher Vector [61] has been shown to outperform standard vector quantization methods such as k-means in the context of Bag-of-Words. On the other hand, the recently proposed VLAD descriptors [35] can be seen as a non-probabilistic version of Fisher Vectors which are faster to compute [35, 64]. In this work we evaluate the accuracy/efficiency trade-off using Fisher Vector and VLAD in the context of video classification.

2.3 Bag-of-Words for video

In this section we explain in detail the pipeline that we use. We mostly use off-the-shelf yet state-of-the-art components to construct our Bag-of-Words pipeline, which is necessary for a good evaluation. Additionally, we explain how to create a fast implementation of densely sampled HOG and HOF descriptors, and also implicitly for MBH, being MBH based on HOG and Optical Flow. We make the HOG/HOF/MBH descriptor code publicly available.

2.3.1 Descriptor extraction

In this section we describe the details of our implementation for dense extraction of HOG, HOF and MBH descriptors. Specifically, in Section 2.3.1 we show how HOG and HOF can be efficiently extracted and aggregated from video blocks. Then, in Section 2.3.1 we deal with MBHs, which are largely based on HOG. Finally, since in this work we compare our implementation with the widely used available code of Laptev [46], in Section 2.3.1, we show the parameters we have adopted in using Laptev's code. Both ours and the Laptev' system work on grey-values only. Note that Laptev's implementation does not include MBH descriptors, thus the comparison performed in our experiments only concerns HOG and HOF.

Fast dense HOG/HOF descriptors

For both HOG and HOF descriptors, there are several steps. First one needs to calculate either gradient magnitude responses in horizontal and vertical directions (for HOG), or optical flow displacement vectors in horizontal and vertical directions (for HOF). Both result in a 2-dimensional vector field per frame. Then for each response the magnitude is quantized in o orientations, usually $o = 8$. Afterwards, one needs to aggregate these responses over blocks of pixels in both spatial and temporal directions. The next step is to concatenate responses of several adjacent pixel blocks. Finally, descriptors have to be normalized and sometimes PCA is performed to reduce their dimensionality, often leading to computational benefits or improved accuracy.

To calculate gradient magnitude responses we use HAAR-features. These are faster to compute than Gaussian Derivatives and have proven to work better for HOG [15]. Quantization in o orientations is done by dividing each response magnitude linearly over two adjacent orientation bins.

We use the classical Horn-Schunk [32] method for optical flow responses as a default. We use the version implemented by the Matlab Computer Vision System Toolbox. Additionally, we evaluate four other optical flow methods: Lucas-Kanade [49], also using the Matlab Computer Vision System Toolbox, the method of Färneback [28], using OpenCV[3] with the mexopencv interface[4], Brox 04 [8], and Brox 11 [9] using the author's publicly available code.

Both HOG and HOF descriptors are created out of blocks. By choosing the sampling rate identically to the size of a single block, one can reuse these blocks. Figure 2.2 shows an example on how a video volume can be divided into blocks. Once responses per block are computed, descriptors

[3] `http://opencv.org`

[4] `https://github.com/kyamagu/mexopencv`

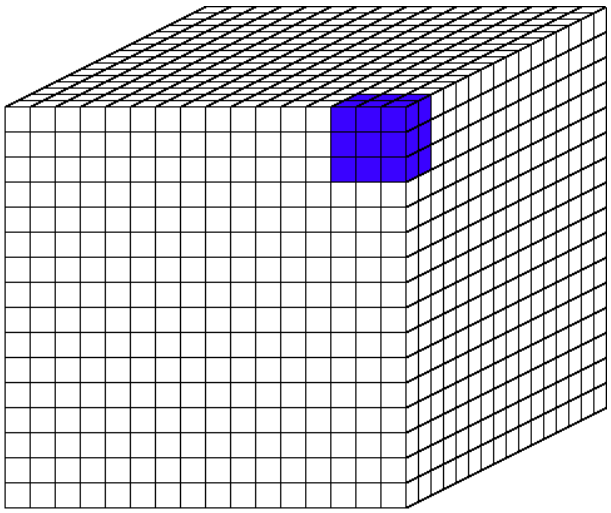

Figure 2.2: Blocks in a video volume can be reused for descriptor extraction. In our work descriptors consist of 3 by 3 blocks in space and 2 blocks in time, shown in blue.

can be formed by concatenating adjacent blocks. In this work we use descriptors of 3 by 3 blocks in the spatial domain and 2 blocks in the temporal domain, as shown in blue in Figure 2.2, but these parameters can be easily changed. Hence each block is reused 18 times (except for the blocks on the borders of the video volume).

To aggregate responses over space we use the Matlab-friendly method proposed by [83]: Let R be an $N \times M$ matrix containing responses in a single orientation (be it gradient magnitude or optical flow magniture). Let B_N and B_M be the number of elementary blocks from which HOG/HOF features are composed. Now it is possible to construct (sparse) matrices O and P of respectively $B_N \times N$ and $M \times B_M$ such that $ORP = A$, where A is a $B_N \times B_M$ matrix containing the aggregated responses for each block. O and P resemble diagonal matrices but are rectangular and the filled in elements follow the 'diagonal' of the rectangle instead of positions (i, i). By proper instantiation of these matrices we perform interpolation between blocks, which provides the descriptors some translation invariance. For integration over time we add the responses of the frames belonging to a

single block. For more details we refer the reader to the work of [83].

In this work, we extract descriptors on a single scale where blocks consist of 8 by 8 pixels by 6 frames, which at the same time is our dense sampling rate. Descriptors consist of 3 by 3 by 2 blocks. Both for HOG and HOF the magnitude responses are divided into 8 orientations, resulting in 144 dimensional descriptors. PCA is performed to reduce the dimensionality by 50% resulting in 72 dimensional vectors. Afterwards, normalization is performed by the L1-norm followed by the square root, which effectively means that Euclidean distances between descriptors in fact reflect the often superior Hellinger distance [1].

Motion Boundary Histograms descriptor

Another commonly used descriptor for video classification tasks is Motion Boundary Histogram (MBH), proposed by Dalal et al. [16], who proved its robustness to camera and background motion. The intuitive idea of MBH is to represent the oriented gradients computed over the vertical and the horizontal optical flow components. The advantage of such representation is that constant camera movements tend to disappear and the description focuses on optical flow *differences* between frames (*motions boundaries*).

In more detail, the optical flow's horizontal and vertical components are separately represented using two scalar maps, which can be seen as gray-level "images" of the motion components. Histograms of oriented gradients are then computed for each of the two optical flow component images, using the same approach used for computing HOG in still images. Taken into account only flow differences, the information about changes in motion boundaries is kept and the constant motion information is removed, which leads to the cancelation of most of the effects of camera motion.

In our MBH implementation we follow the pipeline suggested in [16] and mentioned above. Once computed the horizontal and vertical optical flow

components, histograms of oriented gradients are computed on each image component using the same efficient approach and the same parameters shown in Section 2.3.1. Also the block-based aggregation step is analogous to what described in Section 2.3.1.

The outcome of this process is a pair of horizontal (MBHx) and vertical (MBHy) descriptors [16], each one composed of 144 dimensions. We separately apply PCA to both MBHx and MBHy and we obtain two vectors of 72 dimensions each. The (PCA-reduced) MBHx and MBHy vectors can then be either separately used in the subsequent visual word assignment and classification stages (Figure 2.1) or combined in order to get a unique descriptor. In [86] the authors state that late fusion of MBHx and MBHy gives a better performance than concatenating the two descriptors before the visual word assignment step. Hence in this work we will report results for MBHx and MBHy separately, and a late fusion of the two which we simply denote as MBH. This late fusion combines the outcomes of the two (independent) classifications with equal weights. Finally, in Section 2.4.6, we will also show results concerning a late fusion strategy involving all the descriptors (MBHx, MBHy, HOG and HOF).

Existing HOG/HOF descriptors

We use the existing implementation of Laptev et al. [46]. We use the default parameters as suggested by the authors, which compared to our descriptors are as follows: They perform a dense sampling at multiple scales. At the finest scale, blocks are 12 by 12 pixels by 6 frames, sampling rate is every 16 pixels by every 6 frames. They consider 8 spatial scales and 2 temporal scales for a total of 16 scales, where each scale increases the descriptor size by a factor of $\sqrt{2}$. In the end, they generate around 33% less descriptors than our single scale dense sampling method.

Unlike our descriptor extraction, the implementation of [46] uses 4 ori-

entations for HOG and 5 orientations for HOF, resulting in respectively 72 and 90 dimensional descriptors.

2.3.2 Visual word assignment

We use five different ways of creating a single feature representation of a set of descriptors extracted from a single video: k-means, hierarchical k-means, Random Forests [7, 31], VLAD [35] and Fisher Vectors [61].

For hierarchical k-means we use the implementation made available by VLFeat [84]. For the regular k-means assignment, we make use of the fact that the descriptors are L2-normalised: Euclidean distances are proportional to dot products (cosine of angles) between the vectors. Hence finding the minimal euclidean distance is equivalent to finding the maximal dot product, yet more efficient to compute [83]. For both hierarchical k-means and regular k-means, we use 4096 visual words. For hierarchical k-means, we learn a hierarchical tree of depth 2 with 64 branches per node of the tree (preliminary experiments showed a large decrease in accuracy when using a higher depth with fewer branches, but only marginal improvements in computational efficiency, data not shown). We normalize the resulting frequency histograms using the square root, which discounts frequently occurring visual words, followed by L1-normalization.

Random Forests are binary decision trees which are learned in a supervised way by randomly picking several descriptor dimensions at each node with several random thresholds and choose the one with the highest Entropy Gain. We follow the recommendations of [83], using 4 binary decision trees of depth 10, resulting in 4096 visual words. The resulting vector is normalized by taking the square root followed by L1.

The Fisher Vector [33] as used in [61] encodes a set of descriptors D with respect to a Gaussian Mixture Model (GMM) which is trained to be a generative model of these descriptors. Specifically, the set of descrip-

tors is represented as the gradient with respect to the parameters of the GMM. This can be intuitively explained in terms of the EM algorithm for GMMs: Let G_λ be the learned GMM with parameters λ. Now use the E-step to assign the set of descriptors D to G_λ. Then the M-step yields a vector F with adjustments on how λ should be updated to fit the data (i.e. how the GMM clusters should be adjusted). This vector F is exactly the Fisher Vector representation. We follow [61] and normalize the vector using a square root of the absolute values and afterwards keep the original sign $((sign(f_i))\sqrt{|f_i|})$, followed by L2. In this work we use two common cluster sizes for the GMM: 64 and 256 clusters [61]. Without a spatial pyramid [47], for our 72 dimensional HOG/HOF/MBHx/MBHy features this will yield vectors of 9,216 and 36,864 dimensions respectively. While not comparable with the dimensionality of other methods, Fisher Vectors (and VLAD) allow for linear Support Vector Machines rather than Histogram Intersection or χ^2-kernels. Hence efficiency-wise, the simpler classifiers will compensate for the larger dimension of the feature vectors.

The recently proposed VLAD [35] representation can be seen as a simplification of the Fisher Vector [35, 64] in which: (1) a spherical GMM is used, (2) the soft assignment is replaced with a hard assignment and (3) only the gradient of G_λ with respect to the mean is considered (first order statistics). This leads to a lower dimensional representation, half of the dimensions of a Fisher Vector, in which second order statistics are also used. Following [35] we use for VLAD the same normalization scheme used for Fisher Vectors: We square-root the VLAD vectors while keeping their sign, followed by L2-normalisation. For good comparison to the Fisher Vectors, we use a dictionary of 128 and 512 clusters respectively, leading to features of dimensionality identical to the Fisher Vectors: 9,216 and 36,864 dimensions.

We use the Spatial Pyramid [47] in all our experiments. Specifically, we

divide each video volume into the whole video and into three horizontal parts which intuitively roughly corresponds to a ground, object, and sky division (in outdoor scenes).

2.3.3 Classification

For classification we use Support Vector Machines which are powerful and widely used in a Bag-of-Words context (e.g. [14, 47, 83, 84]). For k-means, hierarchical k-means, and Random Forests, we use SVMs with the Histogram Intersection kernel, using the fast classification method as proposed by [50]. For the Fisher Vector and VLAD, we use linear SVMs. For both types of SVMs, we make use of the publicly available LIBSVM library [11] and the fast Histogram Intersection classification of [50].

2.4 Experiments

Our baseline consists of densely sampled HOG, HOF and MBH(x/y) descriptors, all consisting of blocks of 8 by 8 pixels by 6 frames. For HOF and MBH(x/y), optical flow is calculated using Horn-Schunk. Gradient and flow magnitude responses are quantized in 8 bins. The final descriptors consist of 3 by 3 by 2 blocks. PCA always reduces dimensionality of descriptors by 50%. We use a spatial pyramid division of $1 \times 1 \times 1$ and $1 \times 3 \times 1$ [47] (we have no temporal division). Normalisation after word assignment is done by either taking the square root while keeping the sign followed by L2 for the Fisher Kernel, or by the square root plus L1 for all other methods. We use SVMs for classification, with either a linear kernel for the Fisher Vectors or histogram intersection kernel for all other visual word assignment methods.

Starting from our baseline we perform four experiments: (1) We compare five different visual word assignment methods: k-means, hierarchical

k-means, Random Forests, VLAD and the Fisher Kernel; (2) We compare our densely extracted descriptors with the descriptors provided by Laptev et al. [46]; (3) We evaluate the efficiency/accuracy trade-off by subsampling video frames for the descriptor extraction process; (4) For HOF and MBH(x/y) descriptors, we compare five different optical flow implementations: Horn-Schunk, Lukas-Kanade, Farnebäck [28], Brox 04 [8] and Brox 11 [9].

All timing experiments are performed on a single core of an Intel(R) Xeon(R) CPU E5620 2.40GHz. We use mainly Matlab, but most toolboxes used by us have mex-interfaces to c++ implementations for critical functions. All implementations are heavily optimized for speed. Since the computation involves many common operations that use standardized and optimized libraries (e.g. convolutions, matrix multiplications) on large quantities of data, virtually the entire time is spent on core calculations while the overhead is negligible; using only c++ will not result in noticeable differences in the overall timing results presented in this work.

Based on our experiments we provide two recommendations, one for real-time video classification and one for accurate video classification. Finally we give a comparison with the state-of-the-art.

2.4.1 Dataset

We perform all experiments on the UCF50 Human Action Recognition dataset [63]. This dataset contains 6600 realistic videos taken from Youtube and as such has large variations in camera motion, object appearance and pose, illumination conditions, scale, etc. The 50 human action categories are mutually exclusive and include actions such as biking, diving, drumming, and fencing. The frames of the videos are 320 by 240 pixels. The video clips are relatively short with a length that varies around 70-200 frames. The dataset is divided in 25 predefined groups. Following the

standard procedure we perform a leave-one-group-out cross-validation and report the average classification accuracy over all 25 folds. Optimization of the SVM slack parameter is done for every class for every fold on the training set (containing 24 groups).

2.4.2 Visual word assignment

In this experiment we compare the following visual word assignment methods: k-means, hierarchical k-means, Random Forests, VLAD and Fisher Vector. K-means, hierarchical k-means and Random Forests are similar in the sense that the final vector represents visual word counts. To compare these methods we ensure that all have 4096 visual words. For k-means this means performing clustering with k=4096. For hierarchical k-means we use a hierarchy of depth 2 with 64 branches at each node. The Random Forest consists of 4 trees of depth 10. We choose to base our Fisher Vectors on standard sizes for the number of clusters: 64 and 256 clusters [61, 12]. While Fisher Vectors are of higher dimensionality, the vectors work with linear classifiers. This means that Fisher Vectors are best compared with the other visual word assignment methods in terms of the accuracy/efficiency trade-off. Similarly, we adopted 2 standard cluster sizes for VLAD: 128 and 512 dimensions respectively [35] and we used linear classifiers as well.

The accuracy and computational efficiency for the various word assignment methods for our HOG, HOF and MBH(x/y) features are presented in Figure 2.3 and Table 3.2. The first thing to notice is that the Fisher Vector with 256 clusters has the best accuracy of 0.765 for HOG, 0.795 for HOF, 0.796 for MBHx and 0.804 for MBH, while taking 3.39 seconds per video (per descriptor type). K-means has also good accuracy at 0.728 for HOG, 0.791 for HOF, 0.782 for MBHx and 0.8 for MBH. However, the computational time is at 1.81 seconds per video. This means that the Fisher

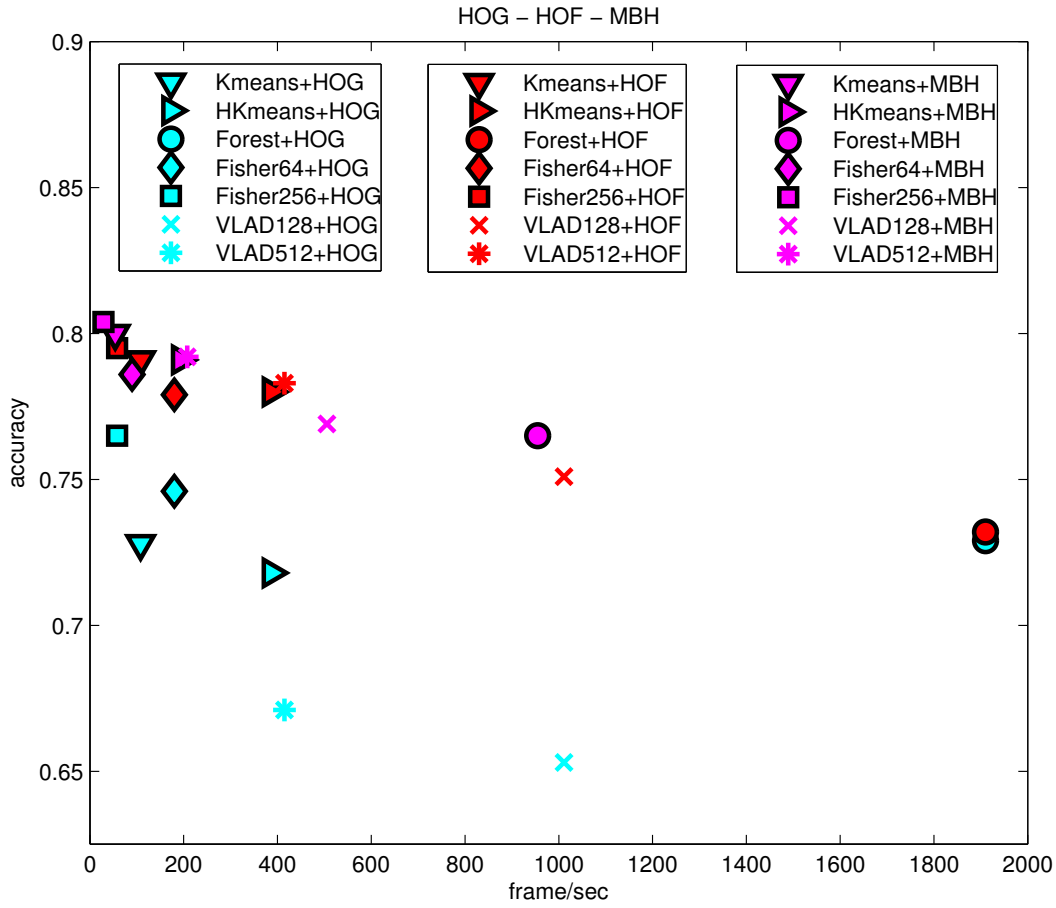

Figure 2.3: Accuracy/Efficiency trade-off for various word assignment methods and features. For a better readability of the figure, we omitted the results concerning MBHx and MBHy (see Table 3.2).

Vector (with 256 clusters) for video classification is superior in accuracy but slightly slower compared to k-means. For computational efficiency, the Random Forest is by far the fastest and takes 0.1 seconds per video. The hierarchical k-means (hk-means) is four times slower at 0.47 seconds per video, and performs slightly worse on HOG (0.718 hk-means vs. 0.729 RF) but significantly better on HOF (0.780 hk-means vs. 0.732 RF) and on MBHx, MBHy and MBH (respectively, 0.774 vs 0.738, 0.763 vs. 0.739 and 0.791 vs 0.765).

	k-means	hk-means	RF	FV 64	FV 256	VLAD 128	VLAD 512
HOG Acc	0.728	0.718	0.729	0.746	0.765	0.653	0.671
HOF Acc	0.791	0.780	0.732	0.779	0.795	0.751	0.783
MBHx Acc	0.782	0.774	0.738	0.767	0.796	0.749	0.774
MBHy Acc	0.772	0.763	0.739	0.759	0.787	0.737	0.765
MBH Acc	0.800	0.791	0.765	0.786	0.804	0.769	0.792
sec/video	1.81	0.51	0.10	1.10	3.39	0.19	0.47
frame/sec	108	387	1910	180	58	1011	415

Table 2.1: Trade-off accuracy/efficiency for the following visual word assignment methods: k-means, hierarchical k-means (hk-means), Random Forest (RF), Fisher Kernel with 64 and 256 clusters (FK 64 and FK 256). Assignment time for HOG and HOF is the same.

In terms of classification time per video, we measure 0.017 seconds per video when using the fast Histogram Intersection based classification for SVMs [50] for k-means, hk-means, and Random Forests. We measure 0.001 seconds per video for the linear classifier used on the Fisher Vector representation with 256 clusters. This means that the classification time is negligible compared to the word assignment time and is of little concern for video classification.

For the remainder of this work, we choose to perform our evaluation on two word assignment methods: the Fisher Vector, which yields the most accurate results, and hk-means, which is the second fastest after Random Forests, while its accuracy for HOF and MBH(x/y) is much higher than using Random Forests.

2.4.3 Comparison with Laptev et al.

In this experiment we compare the publicly available code from [46] with our implementation. We compare only to the dense sampling option as [90] has already proven that dense sampling outperforms the use of space-time interest points. Moreover, only HOG and HOF features are used for com-

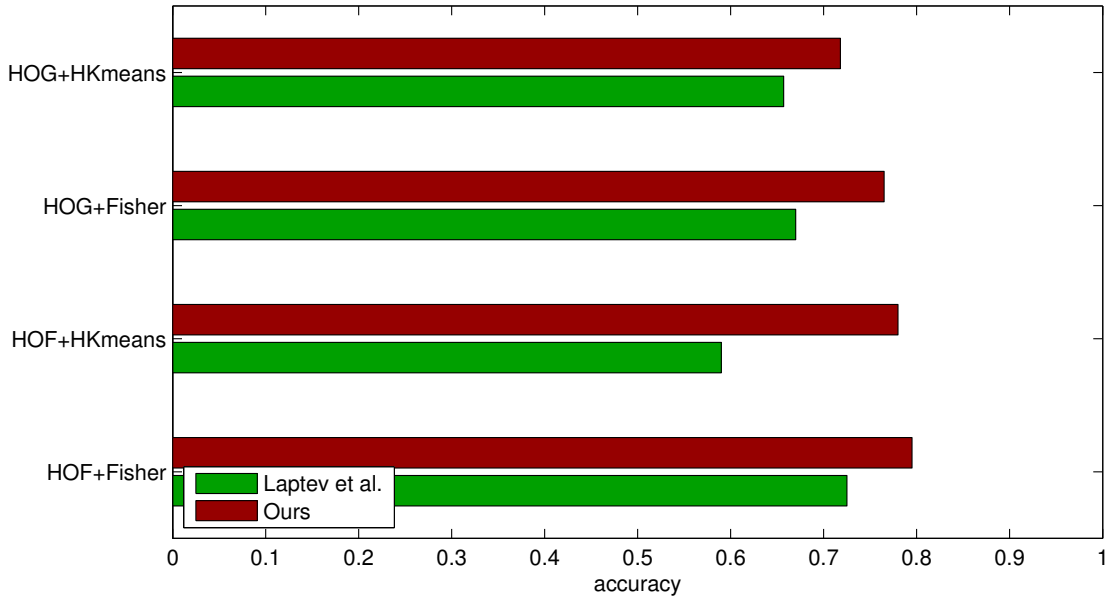

Figure 2.4: Accuracy comparison between [46] and our HOG/HOF descriptors

parison because the code in [46] does not include any implementation for MBH features. Results are presented in Figures 2.4 and 2.5 and in Table 2.2.

	hk-means		FV 256		efficiency	
	HOG	HOF	HOG	HOF	sec/vid	frame/sec
[46]	0.657	0.590	0.670	0.725	141	1.4
ours	0.718	0.780	0.765	0.795	15	12.8

Table 2.2: Comparing the dense HOG/HOF implementation of [46] and ours. The descriptor extraction time is measured for extracting both HOG and HOF features, as the binary provided by [46] does always both. Descriptor extraction time is independent of the visual word assignment method (RF or FV 256).

The results show that for all settings there is a significant difference in accuracy between the dense implementation of [46] and our method. For the Fisher Vector, HOG descriptors yield 0.670 accuracy for [46] and 0.765 accuracy for our implementation and HOF descriptors yield 0.725 accuracy

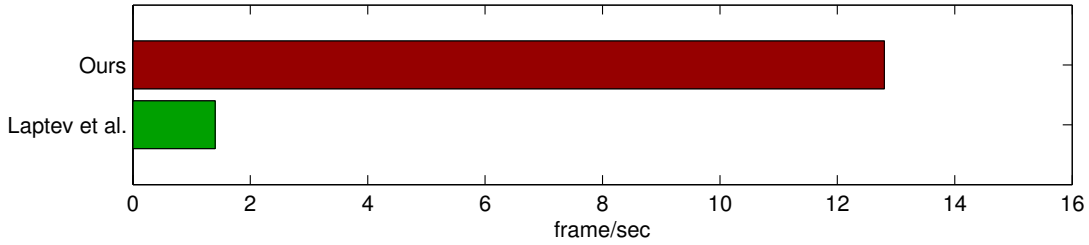

Figure 2.5: Computational Efficiency comparison between [46] and our HOG/HOF descriptors

for [46] and 0.795 accuracy for our implementation. These are accuracy increases of 9% and 7% respectively. Similar differences are obtained using hk-means. Part of the difference can be explained by the fact that we sample differently: because we reuse blocks of the descriptors, our sampling rate is defined by the size of a single block. This means we sample descriptors every 8 pixels and every 6 frames at a single scale, whereas [46] samples every 16 pixels and every 6 frames at 10 increasingly course scales. For our method this yields around 150 descriptors per frame or around 29,000 descriptors per video whereas [46] generates around 90 descriptors per frame or around 17,500 descriptors per video, which means we generate 66% more descriptors. While this may seem unfair towards [46], in this work we are interested in the trade-off between accuracy and computational efficiency, which makes the exact locations from where descriptors are sampled irrelevant.

In terms of computational efficiency our method is more than 9 times faster: their method takes 141 seconds per video while our method takes 15 seconds per video. Our method is faster because we reuse blocks in our dense descriptor extraction method. Note that because the method of [46] samples fewer descriptors, visual word assignment time is faster. But by using [46] the overall computation time will be completely dominated by descriptor extraction.

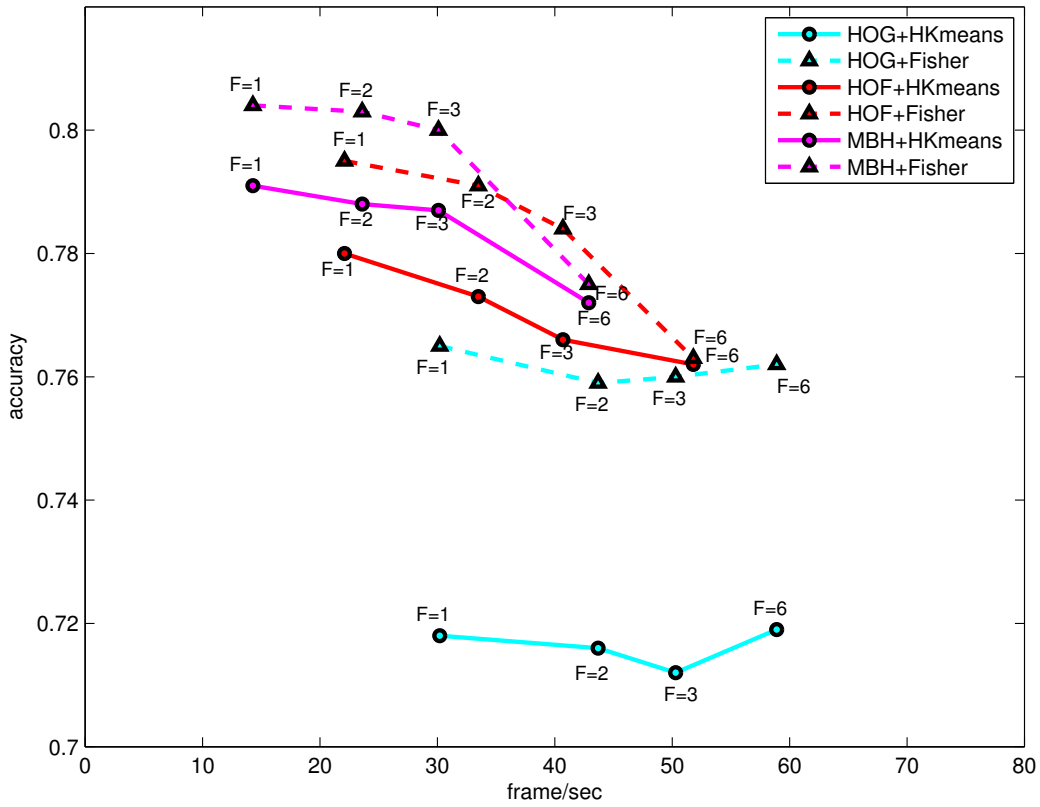

Figure 2.6: Trade-off accuracy/efficiency when varying sampling rate. F stands for frames per block and it is directly related to sampling rate.

To conclude, our implementation is significantly faster and significantly more accurate than the version of [46].

2.4.4 Subsampling video frames

In video, subsequent video frames largely contain the same information. As the time for descriptor extraction is the largest bottleneck in video classification, we investigate how the accuracy behaves if we subsample video frames and hence speed-up the descriptor extraction process.

For a fair comparison, we want the descriptors always to describe the same video volume. In our baseline, each descriptor block consists of 8 by 8

	$\binom{\text{frames/block}}{\text{sample rate}}$	$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
HOG	hk-means	0.718	0.716	0.712	0.719
	FV 256	0.765	0.759	0.760	0.762
	sec/vid	6.5	4.5	3.9	3.3
	frame/sec†	30.2	43.7	50.3	58.9
	$\binom{\text{frames/block}}{\text{sample rate}}$	$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
HOF	hk-means	0.780	0.773	0.766	0.762
	FV 256	0.795	0.791	0.784	0.763
	sec/vid	8.9	5.9	4.8	3.8
	frame/sec†	22.1	33.5	40.7	51.8
	$\binom{\text{frames/block}}{\text{sample rate}}$	$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
MBHx	hk-means	0.774	0.767	0.769	0.758
	FV 256	0.796	0.794	0.788	0.771
	sec/vid	9.4	6.1	5.0	3.9
	frame/sec†	20.9	32.1	39.4	50.7
	$\binom{\text{frames/block}}{\text{sample rate}}$	$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
MBHy	hk-means	0.763	0.757	0.752	0.741
	FV 256	0.787	0.785	0.772	0.750
	sec/vid	9.4	6.1	5.0	3.9
	frame/sec†	20.9	32.1	39.4	50.7
	$\binom{\text{frames/block}}{\text{sample rate}}$	$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
MBH	hk-means	0.791	0.788	0.787	0.772
	FV 256	0.804	0.803	0.800	0.775
	sec/vid	13.7	8.3	6.5	4.6
	frame/sec†	14.3	23.6	30.1	42.9

Table 2.3: Trade-off between frame sampling rate and accuracy. We keep video volumes from which desciptors are extracted the same for all sampling rates. †Frames/second is measured in terms of the total number of frames of the video, not in terms of how many frames are actually processed during descriptor extraction.

pixels by 6 frames. To subsample in such a way that every block describes the same video volume regardless of the sampling rate, we do the following: if we sample every 2 frames, we aggregate responses over 3 frames (i.e. of frame 2, 4 and 6). When sampling every 3 frames, we aggregate responses over 2 frames (i.e. frame 2 and 5), and when sampling every 6 frames in which we only consider a single frame per descriptor block (i.e. frame 3). Results are presented in Figure 2.6 and Table 3.7.

For HOG descriptors, subsampling video frames has surprisingly little effect on the accuracy, both for hk-means and Fisher Vectors: using Fisher Vectors, a sampling rate of 1 yields an accuracy of 0.765 while a sampling rate of 6 yields 0.762 accuracy. The result of hk-means is basically constant, with slight oscillations. In terms of computational efficiency, a significant speed-up is achieved: sampling every 6 frames instead of every frame gives a speed-up from 6.5 seconds per video to 3.3 seconds per video.

For HOF descriptors, subsampling has a bigger impact: For the Fisher Vector accuracy is 0.795 using a sampling rate of 1, maintains a respectable 0.791 accuracy at a subsampling rate of 2 frames, while dropping significantly to 0.763 for sampling every 6 frames. Accuracy with hk-means is less affected and drops from 0.78 at sample rate of 1 to 0.762 at sample rate 6. Again, a good speed-up is obtained by subsampling. While descriptor extraction takes 8.9 seconds when using every frame, a sampling rate of 2 yields a factor 1.5 speed-up while sampling every 6 frames yields a factor 2.34 speed-up.

The remaining rows of Table 3.7 present results obtained with different combinations of the vertical and the horizontal components of the Motion Boundary Histograms (Section 2.3.1). Note that when calculating both components of the MBH features, the optical flow has to be calculated only once, so computation time is faster than simply adding the times of MBHx and MBHy.

We observe a particular order of accuracy among these three combinations: using the only horizontal component (MBHx) always results in a higher accuracy than using the only vertical component (MBHy), independently of whether Fisher Vectors or hk-means is used as word assignment method. This sharp difference is probably due to the fact that in the test videos the horizontal motion is more frequent than the vertical one. Moreover, as expected, late fusion of the two components (MBH), always outperforms using MBHx only. Concerning the drop of accuracy depending on the sample rate, for all the three descriptor combinations (MBHx, MBHy, MBH) and both word assignment methods (Fisher Vectors and hk-means), the accuracy loss as a function of the sample rate is similar to what happens with HOF and much higher than HOG. We believe that this is due to the fact that HOG are basically "static" features, representing the appearance of a given image window independently of possible motion information. As a consequence, they are less affected by optical flow errors (which is used to compute both HOF and MBH(x/y)) and better exploit the redundancy of consecutive video frames.

As for HOG and HOF and also for MBH(x/y) and MBH, we observe a significant computational efficiency gain using subsampling. For instance, sampling every 6 frames yields a factor of 2.4 speed-up for MBH(x/y) and a factor of 3 speed-up for MBH with respect to using all the frames.

To conclude, HOG descriptors can be sampled every 6 frames with negligible loss of accuracy yielding a speed-up of a factor 2. HOF and MBH descriptors can be sampled every 2 frames with negligible loss of accuracy yielding a speed-up of a factor 1.5 and 1.7 respectively. When speed is more important than accuracy, both HOF and MBH descriptors can also be sampled every 6 frames leading to 1-3% accuracy loss while gaining a significant speed-up of a factor 2.3-3.

2.4.5 Choice of Optical Flow

The results reported in the previous section show that both the HOF and the MBH(x/y) descriptors are much more expensive to extract than the HOG descriptors (Table 3.7). This is because calculating the optical flow is computationally expensive. Additionally, not much research has been done on how different optical flow methods affect HOF/MBH descriptors. Therefore in this experiment we evaluate five available optical flow implementations to investigate both their computational efficiency and accuracy. In particular, we compare: (1) Farnebäck [28] from OpenCV using the mexopencv interface, (2) Lucas-Kanade [49] and (3) Horn-Schunk [32] from the Matlab Computer Vision Systems Toolbox, (4) Brox 04 [8] and (5) Brox 11 [9] using the available author's code.

Results are presented in Tables 2.4 and 2.5. Specifically, while in Table 2.4 we used the same setting adopted in the other experiments of this work, in Table 2.5 we downscaled the frame resolution of all the videos by a factor of 4 (i.e., using 80×60 pixel frames) and we subsampled every 6 frames (see Section 2.4.4). This scale and time subsampling was necessary in order to process our large video dataset with both Brox 04 and Brox 11, two state-of-the-art dense optical flow methods not able to process videos in real time. In fact, processing all the frames of our 6600 videos at full spatial resolution with Brox 11 would require a few months.

With the original frame resolution (Table 2.4), and with both hk-means and Fisher Vectors, the three computationally feasible optical flow methods have the same ranking in terms of accuracy. For the Fisher Vector, Horn-Schunk performs best at an accuracy of 0.795, followed by Lucas-Kanade at an accuracy of 0.747, while the method of Farnebäck performs relatively poorly with an accuracy of 0.641. These results show that the optical flow method is crucial to the performance of the HOF descriptor: the choice of

	Horn-Schunk	Lucas-Kanade	Färneback
hk-means	0.780	0.750	0.652
FV 256	0.795	0.747	0.641
sec/video	8.8	7.2	19.0
frame/sec	22	27	10

Table 2.4: Comparison of different optical flow methods used to compute HOF features. Results obtained with no frame subsampling and at full original spatial resolution (320×240 pixels).

	Horn-Schunk	Lucas-Kanade	Färneback	Brox 04	Brox 11
hk-means	0.713	0.681	0.529	0.548	0.552
FV 256	0.718	0.697	0.542	0.638	0.652
sec/video	2.9	2.8	0.76	7.2	12.4
frame/sec	68	69	257	27.4	16

Table 2.5: Comparison of different optical flow methods used to compute HOF features. Results obtained subsampling a frame every 6 and at reduced spatial resolution (80×60 pixels).

optical flow affects the results by up to 15%(!).

In terms of computational efficiency, Lucas-Kanade is the fastest at 27 frames/second, followed by Horn-Schunk at 22 frames per second, while Farnebäck is slower with 10 frames/second. However, while Lucas-Kanade is faster, its trade-off between efficiency and accuracy is not good: As seen in Table 3.7 Horn-Shunk with a frame sampling rate of 2 outperforms the Lukas-Kanade results in Table 2.4 in both speed (33 frames vs 27 frames) and accuracy (0.77 vs 0.75).

Table 2.5 reports results when we subsample frames and reduce the frame size by a factor 4, enabling comparison with the Brox methods. Note that for a fair comparison these times include the computation for reducing the frame sizes (although these times are negligible compared to the total description extraction time). It can be seen that both Brox methods are better than Farneback, but surprisingly not better than the Horn-Shunk and Lucas-Kanade method. One explanation is that this is due to the low resolution of the frames, which makes dense optical flow extraction not sufficiently accurate. Another possibility is that optical flow methods performing better on optical flow benchmarks are not necessarily optimal for use in classification; reducing mistakes in most parts of the flow may introduce artifacts elsewhere that negatively affect results in a classification framework.

In terms of computational efficiency, Brox 11 is the slowest, followed by Brox 04: even subsampled on reduced frames Brox 04 still processes only 27 frames/sec. In contrast to results without downsampling, Farnebäck is here the fastest method. Apparently, there is some overhead in the Matlab optical flow implementations.

To conclude, the choice of optical flow method drastically influences the power of the resulting HOF descriptor and it is not necessarily correlated with the performance on optical flow benchmarks. Additionally, many op-

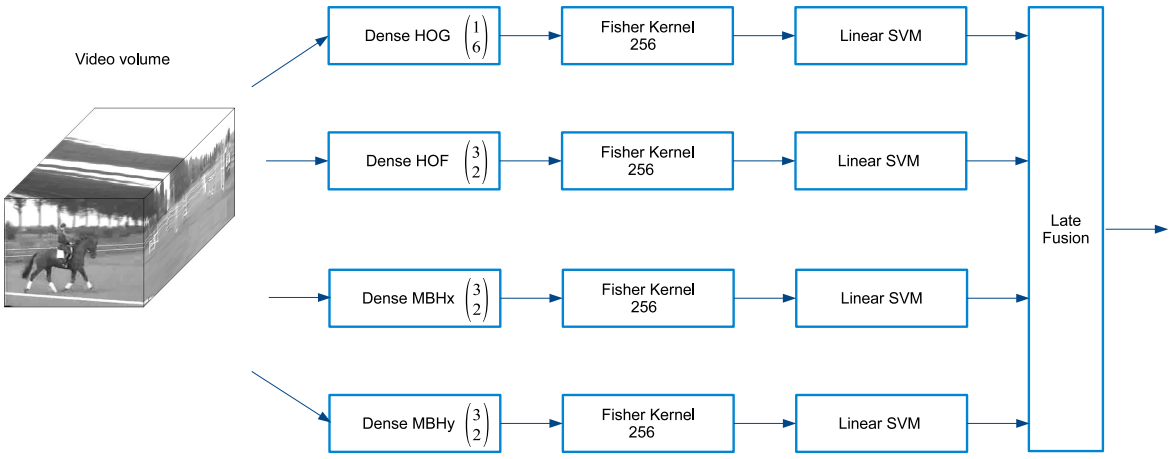

Figure 2.7: Recommended pipeline for accurate video classification. This pipeline yields an accuracy of 0.818 on UCF50 while processing 9 frames per second.

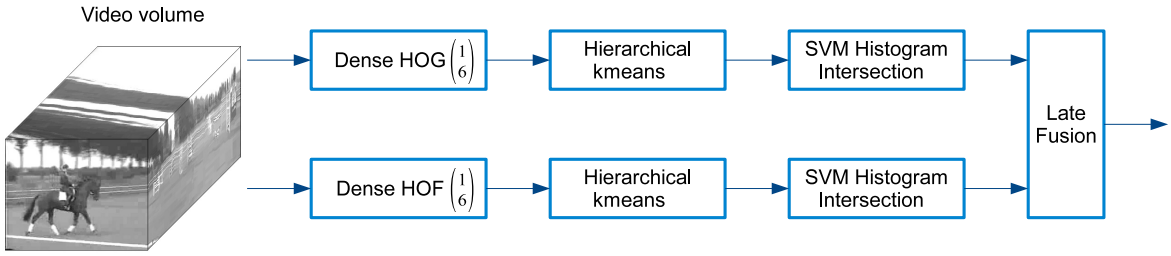

Figure 2.8: Recommended pipeline for realtime video classification. This pipeline yields an accuracy of 0.790 on UCF50 while processing 28 frames per second.

tical flow methods aim for accuracy rather then computational efficiency (e.g. Sun et al. [77] provide a very good overview for accuracy but do not report computational efficiency). Indeed, except the Horn-Schunk, Lucas-Kanade, and Farnebäck methods we did not find any other freely available optical flow method fast enough for use in our classification pipeline. Our evaluation shows that the Horn-Schunk method has the best trade-off between accuracy and computational efficiency and that subsampling every two frames works better than switching to Lucas-Kanade optical flow. Horn-Schunk is therefore the current method of choice.

2.4.6 Recommendations for practitioners

Based on the results of the previous experiments, we can now give several recommendations when accuracy or computational efficiency is preferred. For calculating Optical Flow, Section 2.4.5 showed that the Matlab implementation of Horn-Schunk is always the method of choice. In terms of frame sampling rate, for HOG descriptors we always recommend a sampling rate of every 6 frames. For HOF descriptor, if one wants accuracy we recommend a sampling rate of every 2 frames and if one wants computational efficiency we recommend a sampling rate of 6. The same holds for MBH(x/y) descriptors. For the word assignment method, the Fisher Vector is the method of choice for accuracy. For computational efficiency there are two candidates: hierarchical k-means and the Random Forest. Observe first that the descriptor extraction time is the most costly phase of the pipeline: Extracting HOF descriptors with a sampling rate of 6 frames takes 3.8 seconds per video to compute. And while the Random Forest is five times faster than hierarchical k-means, the difference is only 0.41 seconds per video, which is very small compared to the descriptor extraction phase. Furthermore, Table 3.2 showed a significant drop of accuracy from 0.780 for hierarchical k-means to 0.732 for Random Forests (and a similar drop of accuracy is observed with MBH(x/y)). Therefore we recommend using hierarchical k-means for a fast video classification pipeline.

We found that late fusion of the classifier outputs gave slightly better results than early fusion of the descriptors (e.g. concatenating HOG and HOF). Hence in our recommendations we perform a late fusion with equal weights.

We tested different descriptor combinations, using equal-weights-based late fusion and with the goal of selecting: (1) the most accurate set of

Method	Accuracy
Wang et al. [86] (2013)	0.856%
This work	0.818%
Reddy et al. [63] (2012)	0.769%
Solmaz et al. [75] (2012)	0.737%
Everst et al. [26] (2013)	0.729%
Kliper-Gross et al. [42] (2012)	0.727%

Table 2.6: Comparison with the State-of-the-Art.

descriptors, possibly taking into account the complementarity of appearance/motion information of different features, and (2) the fastest solution with a sufficiently good accuracy degree. The final recommended pipelines are visualized in Figures 2.7 and 2.8.

The most accurate pipeline (Figure 2.7) combines all the descriptors we adopted in this work: HOG, HOF, MBHx and MBHy. HOG are extracted using all the frames, while HOF and MBH(x/y) are extracted with a sampling rate of 2. The word assignment method used in this case is the Fisher Vector. Using this pipeline we can process 11 frames per second (for video frames of 320 by 240 pixels) at an accuracy of 0.818 on UCF50. Conversely, our recommended pipeline for computational efficiency (Figure 2.8) is based on late fusion of only HOG and HOF, both extracted with a sampling rate of 6 and using hk-means. This second pipeline can process 28 frames per second at a respectable accuracy of 0.790.

2.4.7 Comparison to state-of-the-art

In this section we compare our descriptors to the state-of-the-art. Results of several recent works are given in Table 3.10. This comparison is done in terms of accuracy only, as most compared methods evaluate accuracy only. *This work* in Table 3.10 indicates the late fusion of all the descriptors (HOG, HOF, MBH(x/y)): see Section 2.4.6 and Figure 2.7.

As can be seen, the method of [86] yields the best results. This method is a combination of Dense Trajectories and STIP features [46]. As our results are better than [46], we expect that a combination of dense trajectories with our method would increase results further. In general, our method yields good performance compared to many recently proposed methods, which shows that we provide a strong implementation of densely sampled HOG, HOF and MBH(x/y) descriptors.

2.5 Conclusion

this work presented an evaluation of the trade-off between computational efficiency and accuracy for video classification using a Bag-of-Words pipeline with HOG, HOF and MBH descriptors. Our first contribution is a strong and fast Matlab implementation of densely sampled HOG, HOF and MBH descriptors, which we make publicly available.

In terms of visual word assignment, the most accurate method is the Fisher Kernel. Hierarchical k-means is more than 6 times faster while yielding an accuracy loss of less than 2% and is the method of choice for a fast video classification pipeline. HOG descriptors can be subsampled every 6 frames with a negligible loss in accuracy, while being 2 times faster. HOF and MBH descriptors can be subsampled every 2 frames with negligible loss in accuracy, being 1.5 - 1.7 times faster. When speed is essential, HOF and MBH descriptors may be subsampled every 6 frames.

For the HOF and MBH descriptors, we showed that the choice of optical flow algorithm has a large impact on the final performance. The difference between the best method, Horn-Schunk, and the second best method, Lucas-Kanade, is already 5%, while the difference with Färneback is a full 15%. Brox 04 and Brox 11 are computationally very demanding, and cannot be used in a real time video classification scenario.

Compared to the state-of-the-art, the Dense Trajectory method of [86] obtains better results. Nevertheless, the huge difference for the choice of optical flow methods suggests this would also influence dense trajectories. Furthermore, Dense Trajectories still benefit from a combination with normal HOG, HOF and MBH descriptors [39, 86]. Finally, comparisons with other recent methods on UCF50 shows that we provide a strong implementation of dense HOG, HOF and MBH descriptors to the community.

The next chapter continues the work on descriptor extraction by proposing a new efficient descriptor to capture motion information. Furthermore, the next chapter makes the transition to the next important step in video classification, feature encoding, by proposing an important improvement over the existing encoding method VLAD (Vector of Locally Aggregated Descriptors).

Chapter 3

Efficient Human Action Recognition using Histograms of Motion Gradients and VLAD with Descriptor Shape Information[1]

Feature extraction and encoding represent two of the most crucial steps in an action recognition system. For building a powerful action recognition pipeline it is important that both steps are efficient and in the same time provide reliable performance. This work proposes a new approach for feature extraction and encoding that allows us to obtain real-time frame rate processing for an action recognition system. The motion information represents an important source of information within the video. The common approach to extract the motion information is to compute the optical flow. However, the estimation of optical flow is very demanding in terms of computational cost, in many cases being the most significant processing step within the overall pipeline of the target video analysis application. In

[1]Duta, I.C.; Uijlings, J.R.R.; Aizawa, K.; Hauptmann, A.G.; Ionescu, B. and Sebe, N. "Efficient Human Action Recognition using Histograms of Motion Gradients and VLAD with Descriptor Shape Information". In Multimedia Tools and Applications (MTAP), DOI: 10.1007/s11042-017-4795-6, 2017.

this work we propose an efficient approach to capture the motion information within the video. Our proposed descriptor, Histograms of Motion Gradients (HMG), is based on a simple temporal and spatial derivation, which captures the changes between two consecutive frames. For the encoding step a widely adopted method is the Vector of Locally Aggregated Descriptors (VLAD), which is an efficient encoding method, however, it considers only the difference between local descriptors and their centroids. In this work we propose Shape Difference VLAD (SD-VLAD), an encoding method which brings complementary information by using the shape information within the encoding process. We validated our proposed pipeline for action recognition on three challenging datasets UCF50, UCF101 and HMDB51, and we propose also a real-time framework for action recognition.

3.1 Introduction

Over the recent years an explosive growth in video content has occurred and continues growing. As an example of this fulminant increase, Cisco forecast[2] mentioned that the IP video would account for 80% of all IP traffic by 2019. With this huge amount of multimedia content, computational efficiency has become as important as the accuracy of the techniques.

Even though in the past several years there has been an important progress in video analysis techniques, in particular on improving the accuracy of human action recognition in videos [86, 88, 70, 82, 51, 52, 87], the current methods in terms of computational time are able to run with 1-3 frames per second. For instance, in [82] is reported that the popular approach in [45] runs with 1.4 frames per second. Fast video analysis is important in many applications and this issue of efficiency became very

[2] http://newsroom.cisco.com/press-release-content?articleId=1644203

important for large-scale video indexing systems or automatic clustering of large video collections.

The Bag of Visual Words (BoVW) framework with its variations [46, 86, 88] has been widely used and showed its effectiveness in video analysis challenges. The schematic view for a BoVW pipeline is represented in Fig. 3.1, which contains in general three main steps: feature extraction, feature encoding and classification. In addition to these main steps, the framework contains some pre/post processing techniques, such as PCA, feature decorrelation and normalization, which can influence considerably the performance of the pipeline. The commonly used approach for classification is employing a fast SVM classifier over the resulted video representations. The encoding step creates a final representation of the video and a very widely used approach is counting the frequency of the visual words. However, recently super-vector based encoding methods, such as Vector of Locally Aggregated Descriptors (VLAD) [36] and Fisher Vector (FV) [61], obtained state-of-the-art results for many tasks.

The video contains two important sources of information: the static information in the frames and the motion between frames. The feature extraction step focuses mainly on these two directions. The first direction has the goal to capture the appearance information in frames, such as Histogram of Oriented Gradients (HOG) [15, 46]. The other direction is based on optical flow fields like Histogram of Optical Flow (HOF) [46] and Motion Boundary Histograms (MBH) [16]. These descriptors are extracted and combined using Space Time Interests Points (STIP) [45], dense sampling [90, 82] or extracting the descriptors along some trajectories [78, 86, 88].

The pipeline in Fig. 3.1 represents also the common main phases for an action recognition framework. For the classification part, the used approaches are already mature, i.e., most of the existing works, such as [86, 88, 81, 58, 82], use linear SVM, as this is a very fast and effective

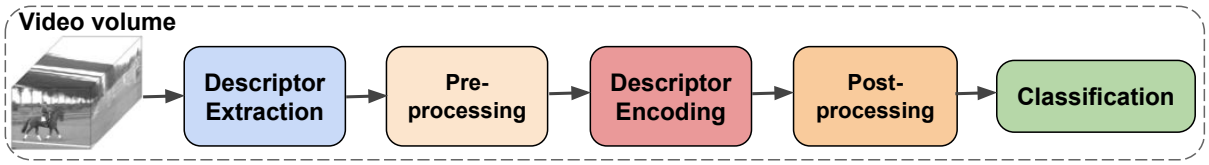

Figure 3.1: The general pipeline for video classification.

method. However, for descriptor extraction and encoding there is still room for improvement. For an efficient video classification system it is necessary that both, descriptor extraction and encoding, to be efficient, otherwise if one of them is not competitive regarding the performance, then the target cannot be reached. As one of the goals of this work is to provide a very efficient system for video classification, we propose new solutions for both steps: descriptor extraction and encoding.

Temporal variation within the videos provides an important source of information about its content. Usually, the temporal information is computed with an optical flow method. There is a large number of approaches for extracting the optical flow fields, from relatively classic methods, such as [49, 32] to relatively recent approaches like [28, 8, 97, 9], which use complex algorithms to compute the motion information. The main drawback of those methods is the high computational cost. This shortcoming becomes the bottleneck in many applications. For instance, the authors in [86] report that optical flow takes more than 50% of the total time for feature extraction. We present in this work a new efficient descriptor, called Histograms of Motion Gradients (HMG), which is based on the motion information. The proposed HMG descriptor captures the motion information using a very fast temporal derivation, which enables us to have similar computational cost as HOG but with a significant improvement in accuracy.

The final representation of the video is one of the key factors for visual recognition such as human action recognition. We can see that in most

of the research works in computer vision and multimedia [88, 58, 82] the super vector-based encoding methods are shown to outperform the other encoding methods. Vector of Locally Aggregated Descriptors (VLAD) [36] is one of the most popular and efficient super vector-based encoding methods which proved its efficiency in creating the final representation of a video for action recognition tasks. Besides its performance, VLAD has several drawbacks. It considers only the mean to represent a cluster of features and also keeps only the first-order statistics and ignores other source of information. The mean is not enough to reflect a distribution, but in general, the mean and the standard deviation can be enough to capture the statistics. To address this issue, this work proposes to improve VLAD by keeping the standard deviation information and incorporating shape information as the difference of standard deviations between the altered standard deviation of local descriptors and the standard deviation of the visual word. This new encoding method, Shape Difference for VLAD (SD-VLAD), captures the distribution shape of the features and brings complementary information to the original VLAD.

The main contributions of this work can be summarized with the following:

- We introduce a new descriptor (HMG), which captures the motion information using a simple temporal derivation, without the need of using the costly optical flow. We make the code for descriptor extraction available[3];

- We propose a new encoding method (SD-VLAD), which captures shape information within the encoding process, providing the best trade-off between accuracy and computational cost. We make the code for descriptor encoding available[3];

[3] `https://iduta.github.io/software.html`

- We adopt several speed-ups, such as fast aggregation of gradient responses, reuse subregions of aggregated magnitude responses, and frame subsampling, which make the pipeline more efficient;

- We propose an integration of our descriptor and encoding method in a specifically designed video classification framework which allows for real-time performance while maintaining the high accuracy of the results.

The rest of the chapter is organized as follows. Section 3.2 presents the related work. Section 3.3 introduces our new proposed descriptor with the adopted approaches for improving the efficiency. The new encoding method is presented in Section 3.5.4. The experimental evaluation and the comparison with state-of-the-art are presented in Section 3.5. Finally, Section 3.6 concludes this work.

3.2 Related work

There are mainly two directions to extract features from a video: hand-crafted and deep learning. One of the state-of-the-art approaches in the hand-crafted category is represented by Improved Dense Trajectory (IDT) [88], where the main goal is to track some points through the video and to extract different descriptors along the trajectories of the points. The work in[88] is an extension of [86] by using an algorithm to cancel the camera motion to obtain more reliable features. The work in [45] proposes Space Time Interests Points (STIP), it has successfully adapted interest points from the domain of images to the domain of video by extending the Harris detector to space-time interest points. The work in [90, 82, 23] uses dense sampling approach. The authors of [90] evaluated several interest point selection methods and several spatio-temporal descriptors. They

found that dense sampling methods generally outperform interest points, especially on more difficult datasets.

The previously mentioned methods establish the region of extracting for several standard descriptors, such as Histogram of Oriented Gradients (HOG) [15, 46], Histogram of Optical Flow (HOF) [46] and Motion Boundary Histograms (MBH) [16]. The work in [47, 46] considers Spatial Pyramid (SP) approach to capture the information about features location. The works in [38, 62] focus on improving the efficiency of action recognition by exploring different alternatives for the computation of the standard optical flow.

Recently, the approaches based on Convolutional Neural Networks (CNN) [40, 71, 70, 96, 79, 56, 6] have proven to obtain very competitive results compared to traditional hand-crafted methods. In general, for action recognition tasks, these works use the two-stream approach where one network is trained on the static images and another network is trained on the optical flow fields. In the end there is a fusion over the output of both networks to provide the final result. In [25, 20] propose solutions for feature encoding specifically designed for deep features. The work in [17] uses a hybrid representation by combining hand-crafted with deep features and takes advantage of different techniques to boost the performance. The work in [93] is fully based on deep features, modeling long-range temporal structure and using a series of good practices to improve the network performance.

The feature encoding is a very important step for action recognition and influences considerably the performance of the general framework. Vector based approaches showed to be very competitive for this step. The most popular super vector encoding methods are: Fisher Vector (FV) [61], Vector of Locally Aggregated Descriptors (VLAD) [36] and Super Vector Coding (SVC) [99]. FV was initially introduced for large-scale image categorization [61]. This encoding method combines the benefits of generative

and discriminative approaches and aggregates the first- and the second-order statistics. FV is performing a soft assignment which in general gives better performance, however, this affects the computational cost. The work in [43] proposes an extension to Spatial Fisher Vector (SFV) which computes per visual word the mean and variance of the 3D spatio-temporal location of the assigned features. VLAD encoding method can be viewed as a simplification of FV which keeps only first-order statistics and performs hard assignment, which makes it much faster than FV. SVC method keeps the zero-order and first-order statistics, thus SVC can be seen as a combination between Vector Quantization (VQ) [72] and VLAD.

There are many precursors who focus on improving VLAD representation, as this is an efficient super vector based encoding method with very competitive results in many tasks. The work in [52] proposes to use Random Forest in a pruned version for the trees to build the vocabulary and then they additionally concatenate second-order information, similar as in FV. Differently from their approach, in this article we keep k-means as clustering method and incorporate second-order information by difference of standard deviations. Another recent work which boosts the performance of VLAD is presented in [57], where the authors suggest improving VLAD by concatenating the second- and third-order statistics, and using supervised dictionary learning. Our approach is different as we consider additionally only the second-order information and build the dictionary in an unsupervised manner. Furthermore, we have a different definition for the first-order statistics by incorporation in the representation the standard deviation, and also our second-order statistics is different in a main key point that we consider an altered standard deviation of local descriptors by counting on the global mean of the cluster instead of local mean of the descriptors. The work in [22] focuses on improving VLAD by using a double assignment approach, and the work in [21] incorporates within

the encoding process the spatio-temporal information showing a consistent improvement in accuracy.

The work in [2] proposes to use intra-normalization to improve VLAD performance. The impact of this approach is to suppress the negative effect of the high values within the vector, which can dominate the similarity between vectors. The authors propose to L2 normalize the aggregated residuals within each VLAD block. We consider also intra-normalization in our framework. Furthermore, they use vocabulary adaptation as an efficient approach to extend the vocabulary to another dataset. In [1] it is proposed RootSIFT normalization to improve the performance of the framework for object retrieval. This normalization approach is based on the idea to reduce the influence of large bin values, by computing square root of the values.

Inspired by these previous works, in this work we propose a new head-crafted descriptor and an extended version for VLAD. The proposed descriptor, Histograms of Motion Gradients (HMG), is computed by initially extracting the motion information by applying a fast temporal derivation between two consecutive frames, then for the resulted "motion image" we compute the horizontal and vertical gradients. From the obtained gradients we compute the magnitude and the angle, and we apply then the quantization and aggregation step to create the final descriptor for a video. For the encoding method, we propose Shape Difference for VLAD (SD-VLAD) where initially a codebook is learnt with k-means and then we compute the final representation with two formulas, one is based on the residual information and the other is focalized on capturing the information regarding the distribution shape by computing the difference between standard deviations. Both source of information are complementary to each other and their combination boosts the performance of the encoding method while achieving a low computational complexity. Our new approach for feature

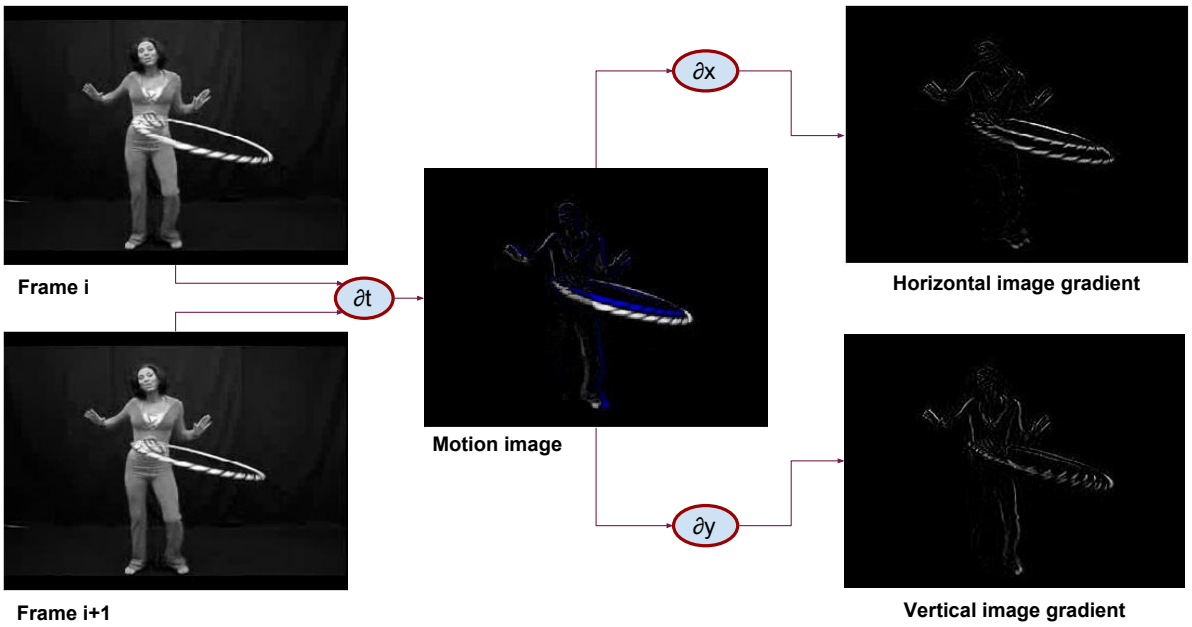

Figure 3.2: Visualization of the process for capturing the motion information for the HMG descriptor. We initially perform a fast temporal derivation over each two consecutive frames, which provides us the motion image. Then we compute the horizontal and vertical gradients for the resulted motion image. The pixels depicted in blue color represent the negative values after temporal derivation.

extraction and encoding allows us to build a very efficient pipeline for video classification, being able to run at more than real-time frame rate.

3.3 Proposed HMG method for descriptor extraction

In this section we introduce the proposed method for capturing motion information from the video. We present several speed-ups that make the framework very efficient, being able to achieve real-time processing.

3.3.1 Histograms of Motion Gradients (HMG)

Our descriptor, Histograms of Motion Gradients (HMG), is based on a temporal derivation to compute the motion information and it is integrated in

the first step of an action recognition framework Fig. 3.1. The illustration of the process of capturing the temporal information is presented in Fig. 3.2. For each two consecutive frames we first compute the temporal derivation:

$$T_{(i,i+1)} = \frac{\partial(F_i, F_{i+1})}{\partial t} \tag{3.1}$$

where F_i is the frame at time index i.

The temporal derivative is computed very effectively by applying a simple and fast filter window [1 -1] for each two consecutive frames (F_i, F_{i+1}). The result of this operation is illustrated in the middle image of Fig. 3.2, where we can observe that the information about the motion between two frames is kept. We can call the output of the applied temporal derivative "motion image". Obviously, after applying the temporal derivation some values are negative, depending on the result of derivation between the pixels in frame i and frame $i + 1$, we represent the negative values with blue color in Fig. 3.2.

After the computation of the temporal derivative, we compute the spatial gradients of the resulted motion image, which allows us to compute the magnitude and the angle of the gradient responses. In the right part of Fig. 3.2 there are represented the horizontal and vertical gradients, computed with:

$$X_{(i,i+1)} = \frac{\partial T_{(i,i+1)}}{\partial x}, \quad Y_{(i,i+1)} = \frac{\partial T_{(i,i+1)}}{\partial y} \tag{3.2}$$

For the computation of spatial gradients we use also the simple and fast filter window [1 0 -1], similar as for HAAR-features. The gradients with this mask are computed much faster than, for instance, Gaussian derivatives. Basically, the gradients with this filter are obtained by making the difference between a frame and its shifted values with one position, once horizontally and once vertically. This makes the computation of gradients very fast.

After we obtain the spatial derivatives, similar as for HOG, we compute the magnitude and the angle:

$$mag = \sqrt{X^2 + Y^2}, \quad \theta = \arctan\left(\frac{Y}{X}\right) \tag{3.3}$$

where each operation from the above formulas is element-wise.

The result of these operations is a 2-dimensional vector field per each new motion frame. We quantize the orientation (θ) in 8 directions/bins and then we accordingly accumulate the magnitude corresponding to each bin. This is similar to how gradient responses are accumulated in SIFT [48]. The next step is to perform the aggregation of those quantized responses over blocks in both spatial and temporal direction. Then we concatenate the responses over several adjacent blocks. We provide in the next subsection the details about the procedure of dividing the video in blocks and volumes. Afterwords, the pipeline in Fig. 3.1 continues with the next step by applying some pre-processing operations before feature encoding, such as normalization and PCA with decorrelation of features. The next steps after the descriptor extraction are very important for the performance of our descriptor. For instance, the descriptors obtained from the motion image may include noise which can result in high peaks that can dominate the entire vector representation. To reduce the negative influence of this aspect, over the initial representation of the descriptors we apply Root-SIFT normalization [1], which penalizes more the high values within the vector, contributing to creating a smoother vector (without large peaks) to represent each local extracted descriptor.

3.3.2 Speed-up HMG extraction

For our proposed descriptor we use a dense sampling strategy to extract the features. In addition to the presented approach for capturing the motion information very efficiently and using fast filters for derivatives, we describe

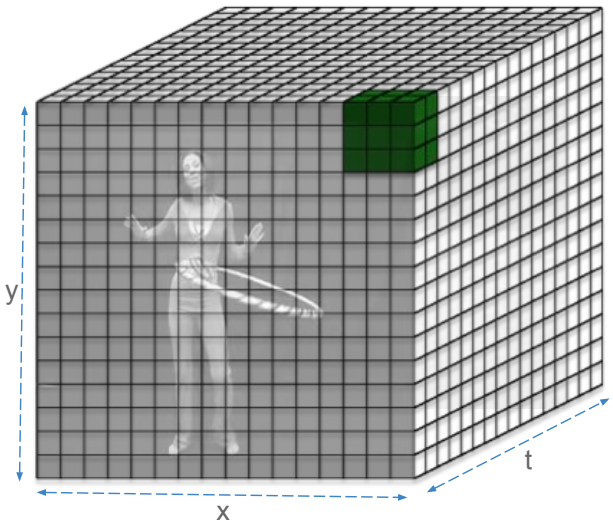

Figure 3.3: The process of dividing the video in blocks and volumes. The part depicted in green represents an illustration of a volume created from 3 by 3 by 2 blocks.

several speed-ups that improve the efficiency of the descriptor extraction process of HMG. The efficiency improvement is performed by taking the advantage of the densely sampled approach and by adopting to our new descriptor several speed-ups presented in [82].

1) Reuse of blocks: Our choice to establish the region of the descriptor extraction is the use of dense sampling strategy since this method has a big potential for efficiency. It can be also easily extended to an even faster version using parallelization. Furthermore, in several works, it has been found to be more accurate than keypoint-based sampling in images [37] and videos [90, 58]. We take advantage of the densely sampled descriptor nature in order to speed up the feature extraction time. Fig. 3.3 illustrates an example for dividing the video into blocks, and how a volume is created of several adjacent blocks. Our HMG descriptor is extracted on a single scale over each block, which consists of 8 by 8 pixels by 6 frames. The size of the blocks is also our dense sampling rate. The green part from the Fig. 3.3 represents a video volume, where the responses over several adjacent blocks are concatenated for creating the final descriptor. Each

video volume consists of 3 by 3 by 2 blocks, corresponding to x, y and t axis. By choosing the sampling rate equally with the block size, then we can reuse the blocks for making the descriptor extraction efficient. Therefore, the representation for a block is computed only once and then use it for the construction of all the volumes around that block. For instance, each block can be reused for 18 times (excepting the blocks on the borders) for the current size of the video volume: 3 by 3 by 2 blocks.

2) Fast aggregation of responses: After we compute the magnitude and the angle, the resulted responses are aggregated for each block. We adopted the approach in [83]. Basically we compute the aggregation of all the frame pixels by doing just a multiplication of three matrices. After the spatial aggregation of 8 by 8 pixels and the temporal aggregation of 6 frames, each block is characterized by 8 values as we consider 8 orientations for quantization of responses. Having 8 bins and a size of 3 by 3 by 2 for video volume, the original dimensionality of our descriptor is therefore 144.

3) Frame subsampling: For efficiency reasons we evaluate HMG by subsampling video frames. Subsequent frames contain redundant information, and the computational cost can be substantially improved by frame subsampling. We evaluate the impact on the accuracy and efficiency of our descriptor by skipping frames. A detailed analysis of the trade-off between accuracy and computational time is presented with the experimental results.

3.4 Proposed SD-VLAD method for descriptor encoding

In this section we present our encoding method for human action recognition. We first review the original VLAD representation.

3.4.1 VLAD representation

VLAD is initially proposed in [36] and can be seen as a simplification of the FV. For the VLAD pipeline first a codebook of k visual words is learned with k-means, $M = \{\mu_1, \mu_2, ..., \mu_k\}$, which are the means for each cluster. For each visual word a subset of local descriptors is assigned based on the nearest neighborhood criterion, $X_i = \{x_1, x_2, ..., x_{n_i}\}$, where x is a feature vector and n_i is the number of assigned features to the i-th visual word. The idea of VLAD is to accumulate for each visual word the residuals (the differences between the assigned descriptors and the centroid):

$$v_i = \sum_{j=1}^{n_i} (x_j - \mu_i) \tag{3.4}$$

The final VLAD representation is a concatenation of all vectors v_i and the final dimensionality of VLAD is $k \times d$, where d is the dimension of the descriptors. The VLAD performance can be boosted by using intra-normalization [2], which normalize independently each VLAD block v_i: $\frac{v_i}{||v_i||_p}$, usually $p = 2$ (i.e., the L2 norm).

3.4.2 Shape Difference for VLAD

The original VLAD is based only on the mean as statistical information, however, for describing a set of descriptors it is more informative to have at least the mean and standard deviation of them. The residuals computed by VLAD algorithm can provide only partial cluster distribution information. Fig. 3.4 shows a case when VLAD fails to provide a good discriminative representation. Even if the centroids μ_1 and μ_2 are completely different and the feature distribution assigned to each cluster differs significantly, the standard VLAD returns the same representation, therefore, the sums over the residuals v_1 and v_2 are completely equal ([1 -8]). Also in the case when the features are distributed in a symmetrical arrangement around

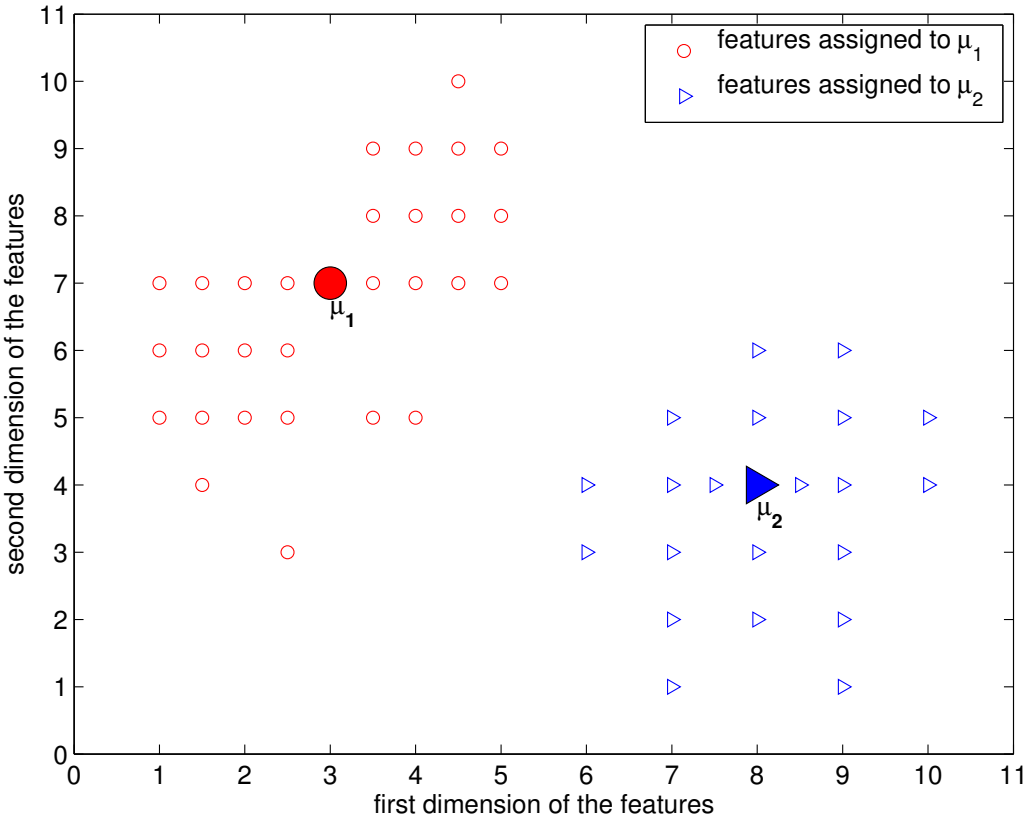

Figure 3.4: An illustrative example when VLAD fails to provide a reliable representation. Each descriptor is assigned to its nearest centroid, μ_1 or μ_2. Even though the distribution of the assigned descriptors to each visual word is completely different, the result of VLAD representation (computed with the standard formula (3.4)) is equal with [1 -8] for both, v_1 and v_2. In this case, only the computation of the residuals is not enough for obtaining a reliable description.

the centroid, then the sum over the residuals is a vector of zeros, this leading to making no difference between the results of a cluster with features distributed symmetrically and a cluster with no features assigned.

For providing a more discriminative representation, it is necessarily to introduce more statistical information. We propose Shape Difference for VLAD (SD-VLAD), which captures information related to the distribution shape of the descriptors. Our final representation is computed with two formulas. Similar to FV, for the first formula, the residuals are divided by standard deviation, and our first part of the final representation is represented as:

$$v_i^{\mu} = \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{x_j - \mu_i}{\sigma_i} \tag{3.5}$$

where n_i is the number of descriptors assigned to the cluster μ_i and σ_i is the standard deviation of the cluster with the mean μ_i.

The division by the number of descriptors, that switches sum pooling to average pooling, is a very simple technique to deal with the problem of burstiness when some parts of VLAD can dominate the entire representation. In the end, these components will have a greater weight for the classifier and influence negatively the performance. This normalization, with the number of descriptors assigned, becomes more important if no intra-normalization technique is used. Intra-normalization is a better strategy to deal with the problem of burstiness, but there are some cases when intra-normalization is not recommended. For instance, in [58] it is underlined that intra-normalization may have a negative effect for sparse features like STIPs.

By considering the normalization (with the number of descriptors as-

signed) for VLAD, the Equation (3.4) becomes:

$$\bar{v}_i = \frac{1}{n_i}\sum_{j=1}^{n_i}(x_j - \mu_i) = \frac{1}{n_i}(\sum_{j=1}^{n_i}(x_j) - n_i\mu_i) = \frac{1}{n_i}\sum_{j=1}^{n_i}(x_j) - \mu_i = \hat{\mu}_i - \mu_i \quad (3.6)$$

where $\hat{\mu}_i$ is the mean of the local descriptors assigned to the cluster μ_i. In this way, VLAD can be seen as the difference between the mean of the local descriptors and its assigned visual word.

To address the shortcoming of VLAD considering only the mean as statistical information, we consider in our representation the shape information. Starting from the Equation (3.6) we can go further with the analogy (the difference between standard deviations) and build the shape difference representation as following:

$$v_i^{\sigma} = \hat{\sigma}_i - \sigma_i = \left(\frac{1}{n_i}\sum_{j=1}^{n_i}(x_j - \mu_i)^2\right)^{\frac{1}{2}} - \sigma_i \quad (3.7)$$

where $\hat{\sigma}_i$ is the altered standard deviation of the local descriptors assigned to cluster μ_i and σ_i is the standard deviation of the cluster μ_i; the power of a vector is the element-wise power. We compute the standard deviation for the local descriptors by using the mean of the cluster and not the local mean of assigned descriptors due to the fact that the local mean of the assigned descriptors may not contain statistical information, as there are many cases when too few descriptors (even one or two descriptors) are assigned to a cluster, especially when the number of clusters is increased. Making the difference of descriptors and their local mean can lead to cases with no information. Instead, by considering the mean of the cluster, which is computed on a large number of descriptors, the difference is more stable, especially for the cases with less descriptors assigned to a cluster.

The shape difference brings complementary information related to the mean, in the experimental part of the chapter we show that it is beneficial for the classifier. For our final SD-VLAD representation we concatenate

the resulting vectors from v^{μ} and v^{σ} (Equation (3.5) and Equation (3.7)). We apply also intra-normalization L2 for each v_i^{μ} and v_i^{σ}.

3.5 Experimental Evaluation

The general pipeline used for evaluation is the one presented in Fig. 3.1, and more details for each step are presented in the remaining part of the chapter.

In the following we present: the datasets used for evaluation (Section 4.5.1); experimental setup (Section 3.5.2); comparison of the proposed descriptor with other dense methods (Section 3.5.3); evaluation of the proposed encoding method together with different standard approaches (Section 3.5.4); comparison of our proposed descriptor with Improved Dense Trajectories approach (Section 3.5.5); the impact of the frame subsampling on the accuracy and on the computational cost (Section 3.5.6); the proposed pipeline for real-time video classification (Section 3.5.7); comparison with the state-of-the-art approaches (Section 3.5.8).

3.5.1 Datasets

We evaluate our framework on three of the most popular and challenging datasets for action recognition: UCF50 [63], UCF101 [76] and HMDB51 [44].

The UCF50 dataset [63] contains 6,618 realistic videos taken from YouTube. There are 50 human action categories mutually exclusive, which range from general sports to daily life exercises. The videos are split into 25 predefined groups. We follow the recommended standard procedure and perform leave-one-group-out cross validation and report average classification accuracy over all 25 folds.

The UCF101 dataset [76] is a widely adopted benchmark for action

recognition, consisting in 13,320 realistic videos, which are divided into 25 groups for each action category. This dataset contains 101 action classes and there are at least 100 video clips for each class. We follow for evaluation the recommended default three training/testing splits. We report the average recognition accuracy over these three splits.

The HMDB51 dataset [44] contains 51 action categories, with a total of 6,766 video clips extracted from various sources, such as Movies, the Prelinger archive, Internet, Youtube and Google videos. It is one of the most challenging dataset with realistic settings. We use the original non-stabilized videos, and we follow the original protocol using three train-test splits [44]. We report average accuracy over the three splits as performance measure.

3.5.2 Experimental setup

For evaluation of our proposed HMG descriptor the baseline is to use dense sampling with 8 by 8 pixels by 6 frames as in [81, 82] and the gradient magnitude quantized in 8 orientations. The final descriptor is a concatenation of 3 by 3 by 2 blocks. For the pre-processing step we perform RootSIFT [1] normalization and then we apply PCA to reduce the dimensionality by a factor of two and decorrelate the features. This yields a final descriptor dimension of 72. We use spatial pyramid in all our experiments, we divide all the frames of the video into three horizontal parts which roughly correspond to a ground, object, and sky division.

The codebook for each experiment needed for feature encoding is built from randomly sampled 500K features of the training set for the specific tested dataset. For the resulted vectors after descriptor encoding we apply power normalization followed by L2 for the super-vector based encoding methods and power normalization followed by L1 for all other visual word assignment methods. The parameter α for power normalization is initially

descriptor	HOG	HOF	MBHx	MBHy	**HMG**
accuracy	0.762	0.799	0.784	0.792	**0.814**
seconds/video	2.67	4.03	4.37	4.37	2.73
frames/second	73.50	48.61	44.80	44.84	71.73

Table 3.1: Accuracy and efficiency comparison of various dense descriptors (results reported on the UCF50 dataset, best result is in bold).

fixed to 0.5. We perform the classification with SVMs, with a linear kernel for super-vector based encoding methods and histogram intersection kernel for all other encoding methods, with $C = 100$.

We initially compare our descriptor with dense HOG, HOF, MBHx and MBHy using the available code from [81, 82]. For these descriptors we use the same settings and speed-ups as presented for HMG, see Section 3.3. The optical flow for HOF, MBHx and MBHy is computed with Horn-Schunck method [32] using the Matlab Computer Vision System Toolbox as recommended in [82]. In [82] is presented a detailed evaluation of using different optical flow approaches with the conclusion that Horn-Schunck method provides the best trade-off between the accuracy and computational efficiency. The timing measurements are performed on a single core Intel(R) Xeon(R) CPU E5-2690 2.60GHz, using 500 randomly sampled videos (10 videos for each class) from the UCF50 dataset. We report the average of the number of seconds per video and the number of frames per second that the system can process. We perform the parameter tuning on the UCF50 dataset.

3.5.3 Comparison to dense descriptors

In this part we present a first comparison between the proposed HMG descriptor and popular dense descriptors for action recognition: HOG, HOF, MBHx and MBHy [82]. The comparison is conducted in terms of accuracy

and computational cost. All the descriptors benefit of the same settings and the same speed-up approaches presented above for the HMG descriptor. All dense descriptors are extracted using only the intensity information. All the computational time measurements for descriptor computation include also the loading time of the video and converting the frames to graylevel. For this set of experiments we use Fisher Vector (FV) [61] as encoding method, with the common setting of 256 clusters. We choose FV for this set of experiments as this is a standard widely used encoding method for action recognition task, thus, the direct comparison with other approaches is straight forward.

The comparative results are presented in Table 3.1. Our approach of computing the motion information by applying a simple and efficient temporal filter does not affect significantly the computational cost as compared with the fast HOG descriptor. While the efficiency is preserved, in terms of accuracy our HMG descriptor outperforms with a large margin HOG, by 5.2 percentage points. This significant performance improvement while preserving the efficiency shows that the motion information captured by our descriptor is very discriminative for videos and can be considered as a good option for the applications based on video analysis, especially for those where the computational cost is crucially important. Remarkably, HMG outperforms even descriptors based on classical optical flow which are more demanding for computational cost. For instance, HMG outperforms HOF by 1.5 percentage points in terms of accuracy, moreover, the descriptor extraction for HMG runs with approximately 72 frames/second while HOF runs only at around 49 frames/second. This big difference in efficiency is due to the optical flow computation, which can take up to 50% of the cost for HOF extraction.

3.5.4 Feature Encoding

In this set of experiments we make the first comparison of our proposed encoding method, SD-VLAD, with the other standard approaches for creating a final representation that serves as input for a classifier.

In this part we compare our dense HMG descriptor with dense HOG, HOF, MBHx and MBHy for Bag-of-Visual-Words (BoVW) using three approaches for visual word assignment: k-means, hierarchical k-means (hk-means) and Random Forests (RF) [7]. In addition we use other three variations of BoVW: Fisher Vectors (FV) [61] and Vector of Locally Aggregated Descriptors (VLAD) [36], and the proposed method Shape Difference VLAD (SD-VLAD). For k-means and hk-means we use the implementation made available with VLFeat [84]. For both we create a codebook of 4,096 visual words. For hk-means we learn a hierarchical tree of depth 2 with 64 branches per node. RF are well-known for their speed, they are binary decision trees, learned in a supervised manner by randomly picking several descriptor dimensions at each node with several random thresholds. The split with the highest Entropy Gain is selected. We follow the recommendations of [83, 81, 82], using 4 binary decision trees of depth 10, which create a codebook of 4,096 visual words.

For FV we keep the codebook size of 256 clusters. We test VLAD representation with 256 and also with 512 visual words for making the comparison between super vector-based encoding methods more fair. For SD-VLAD we fix the codebook size to the standard 256 words. For VLAD and SD-VLAD we apply also intra-normalization L2 as explained in the previous section.

The results for different encoding methods are presented in Table 3.2, which confirm that super-vector encoding methods give a better video representation than the other encoding approaches. The superiority of super-

	k-means	hk-means	RF	FV	VLAD (256)	VLAD (512)	**SD-VLAD**
HOG	0.731	0.720	0.718	0.762	0.712	0.731	**0.768**
HOF	0.789	0.779	0.738	0.799	0.810	0.815	**0.833**
MBHx	0.772	0.760	0.731	0.784	0.782	0.795	**0.800**
MBHy	0.783	0.774	0.750	0.792	0.799	0.814	**0.820**
HMG	0.781	0.759	0.735	0.814	0.805	0.822	**0.834**
sec/video	8.42	0.37	**0.05**	2.09	0.37	0.56	0.38
frame/sec	24	526	**3788**	94	532	352	513

Table 3.2: Accuracy vs. processing time for different encoding methods and using several dense descriptors (results reported on the UCF50 dataset, best results are in bold).

vector encoding methods is due to the fact that they capture information related to the mean and variance/standard deviation of the features and not only the membership information of the features to the clusters. Our HMG descriptor is very competitive for all the encoding methods, especially for super-vector encoding methods, which outperforms all the other descriptors, with an accuracy of 0.834 accuracy for SD-VLAD.

The computational cost for the encoding step is not dependent on the type of features, it depends on the number of visual words and the dimensionality of descriptors. As all our descriptors have the same dimensionality, we reported the computational cost for encoding a descriptor (can be any) with 72 dimensions. The RF approach for encoding step is by far the fastest and takes 0.05 seconds per video, however, the accuracy drops significantly for all descriptors related to the best encoding method for the performance. The results for hk-means represents a good trade-off between accuracy and computational efficiency. It can process the video at a frame rate of 526. When the speed is crucially important then RF is the best choice, encoding the features with 3,788 frames per second.

After these experiments we can take the conclusion that super-vector

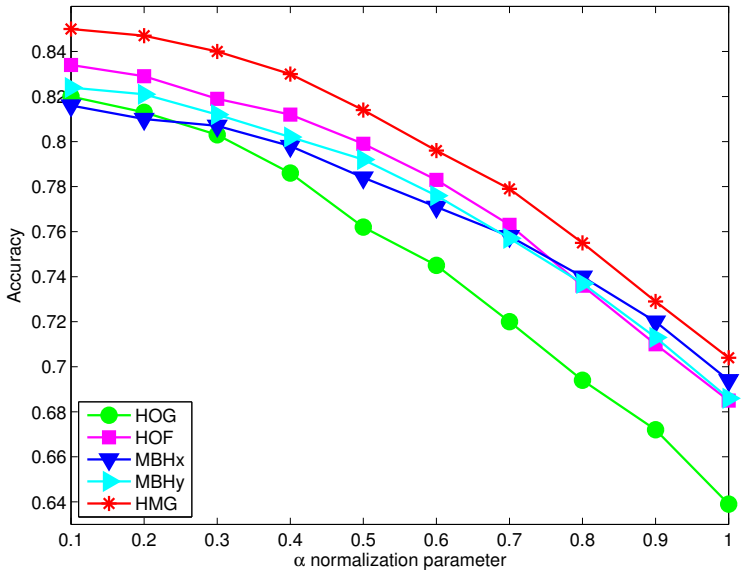

Figure 3.5: Impact of the normalization parameter on the Fisher Vector performance for the UCF50 dataset.

based encoding methods give the best performance. For SD-VLAD the most demanding part for the computational cost is the the codebook assignment and this is the reason that the efficiency of SD-VLAD is similar to the baseline VLAD256. The computation of an extra representation for SD-VLAD does not increase significantly the computational cost. For feature encoding, SD-VLAD represents the best trade-off between accuracy and computational efficiency, running at a frame rate of 513 frames/second. Regarding the encoding method, our goal is to find the best approach for the accuracy of the system and the method with best trade-off between computational cost and accuracy of the pipeline. Considering this, for the further experiments we use only super vector-based encoding methods and we perform some supplementary tests to establish the best approach for the encoding choice.

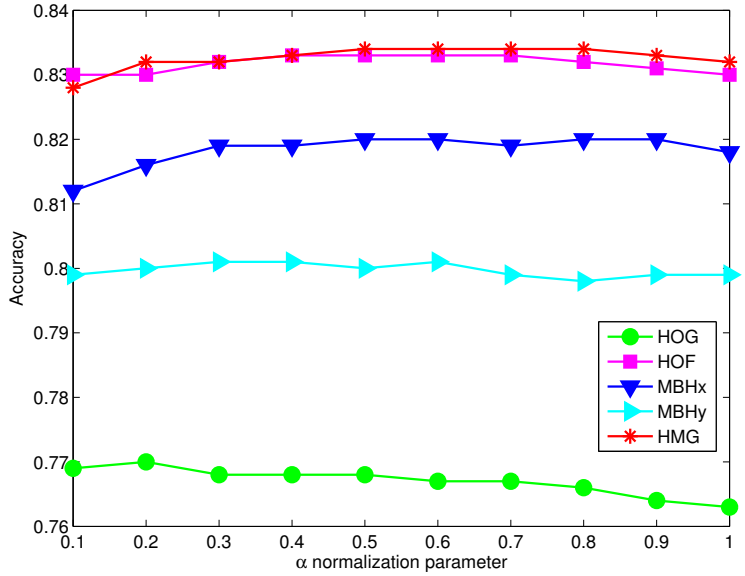

Figure 3.6: Impact of the normalization parameter on the SD-VLAD performance for the UCF50 dataset.

Improving the performance

The post-processing step after the encoding method can boost the performance of the system by preparing the input for the classifier. After feature encoding, for the resulted vector of the video representation we apply before classification Power Normalization followed by L2-normalization ($||sign(x)|x|^{\alpha}||$, where $0 \leq \alpha \leq 1$ is the normalization parameter), we call this PNL2. The effect of this normalization is reducing the peaks within the vector. It is very important for the performance of the system that the values within the vector are not spread on a large interval, since high peaks will dominate the distances between the vectors. As the classifier receives as input the distances between the vectors, the peaks receive a higher weight and the other components of the vector will contribute less, in the end this may influence negatively the classifier output. The α parameter from PNL2 controls the level of penalization, by giving a smaller α, the

large values are shrinked more and reduce the peaks within the vector.

We perform the α parameter tuning within the interval [0.1; 1], with the step 0.1. Fig. 3.5 presents the graphs with the impact of the normalization for FV. We can see that the accuracy is drastically affected by the α normalization parameter for PNL2. There is a continuous increase in performance of all the descriptors when choosing a smaller α. The $\alpha = 1$ actually means that only L2 normalization is applied, and the performance boost between $\alpha = 1$ and $\alpha = 0.1$ is 18.1 percentage points on HOG, 14.9 on HOF, 12.2 on MBHx, 13.8 on MBHy and 14.5 for HMG. This considerable increase in performance for FV when PNL2 is applied with a small α is due to the fact that the resulted final vector after applying the encoding contains large peaks, having a large interval for the values. This is caused by the FV formulation which is built using two different formulas that provide in the end different intervals for the values. PNL2 with a small α helps in bringing the values in a smaller interval, reducing the peaks, and therefore, the distances between vectors are more reliable. For all the next experiments we set $\alpha = 0.1$ for PNL2 when using FV as encoding method.

Fig. 3.6 shows the parameter tuning for PNL2 when using SD-VLAD as encoding method. In this case the influence of α normalization parameter is not that radical. This is due to the fact that we apply intra-normalization L2 when encoding the features with SD-VLAD and therefore, the peaks within the vector are already reduced. However, if intra-normalization is not used during the encoding process then we recommend to use PNL2 with a very small α for the resulted vector, similar as for FV. For all the next experiments we keep the initial setting of PNL2 with $\alpha = 0.5$ for VLAD and SD-VLAD.

Improving the efficiency

From the previous experiments we can take the conclusion that FV and SD-VLAD provide the best results. SD-VLAD is the choice for a trade-off between the computational cost and accuracy. One of the reasons for which VLAD and SD-VLAD are more efficient than FV is the assignment approach. The VLAD based encoding methods use hard assignment while FV uses soft assignment making this step more demanding for the computational cost.

VLAD and SD-VLAD are the choices for a trade-off between accuracy and computational cost. In this set of experiments we investigate how we can improve the computational cost for VLAD and SD-VLAD methods without affecting negatively the accuracy. The assignment step is the most demanding part for the computational time of an encoding method. To decide to which centroid a feature belongs, it is necessary to compute the distance to all centroids of the codebook and to assign the feature to the closer visual word. We evaluated two approaches to compute the distances. First approach is to use the standard Euclidean distance. After we compute the distances we take the minimum value to decide to which visual word the feature belongs. The second approach is to use inner product to compute the distances. By making unit length for both vectors for which we compute the distance we can apply inner product operation as a measurement for the distance. In this case, to decide to which centroid a feature belongs we take the maximum value among the computed inner products. If all feature vectors have the same length (e.g. unit length), then taking the maximum inner product is equivalent to taking the minimum Euclidean distance.

Table 3.3 presents the comparison results between Euclidean distance and the inner product used for the assignment step within the encoding

	VLAD256		VLAD512		**SD-VLAD**	
	E. dist.	inner p.	E. dist.	inner p.	E. dist.	inner p.
HOG	0.712	**0.717**	0.731	**0.735**	0.768	**0.772**
HOF	0.810	**0.812**	0.815	**0.823**	0.833	**0.835**
MBHx	0.782	**0.789**	0.795	**0.801**	0.800	**0.802**
MBHy	0.799	**0.802**	0.814	**0.816**	0.820	**0.822**
HMG	0.805	**0.809**	0.822	**0.823**	0.834	**0.834**
sec/video	0.37	**0.20**	0.56	**0.32**	0.38	**0.23**
frame/sec	532	**971**	352	**620**	513	**848**

Table 3.3: Performance comparison between Euclidean distance (E. dist.) and inner product (inner p.) used for the assignment step during the encoding process (results reported on the UCF50 dataset, best results are in bold).

process. We can see that the encoding method is much faster when using the inner product than in the case of Euclidean distance, being able to improve the computational cost from 513 to 848 frames per second for SD-VLAD. The slight improvement of the accuracy is the effect of applying L2 norm for making unit length when we compute the inner product. For all next experiments we use the inner product for the assignment step when VLAD-based methods are used for encoding.

Closer comparison of SD-VLAD with direct competitors

Our proposed encoding method, SD-VLAD, provides the best trade-off between the accuracy and computational cost. We provide a direct comparison of our method with the closest approaches for the encoding step. The closest approaches to ours are represented by VLAD [36] and the work in [57]. The authors in [57] use high-order statistics for VLAD and dictionary learning to boost the performance. They use three order statistics, which makes the final vector three times bigger than VLAD.

The goal of this comparison is to discover which formulas are better

to compute the final representation that will serve as input for the classifier. Of course, in this comparison all the methods benefit of the same settings including using intra-normalization L2 and inner product for the assignment step. As in the previous experiments, the computational time reported includes also the time for making the unit length for the vectors needed for inner product and also the time for intra-normalization. Furthermore, besides the fact that the vocabulary is build in the same way for all approaches, for VLAD256, H-VLAD [57] and for SD-VLAD we use also the same codebook (this is straight forward as we use for all 256 visual words for the codebook size), which makes the comparison more reliable as the randomness of constructing the codebook is excluded.

Table 3.4 presents the efficiency and performance comparison for all five dense descriptors. In terms of accuracy our approach outperforms the other methods for all descriptors by a large margin. Furthermore, the dimensionality of our final vector is 33% less than the representation in [57], as we concatenate two order information and they concatenate three order statistics. In general, high order information leads to a good improvement compared to the original VLAD. In terms of computational cost, VLAD256 is the fastest. VLAD512 and H-VLAD are almost twice as slow with moderate accuracy improvements. In contrast, SD-VLAD is almost as fast as VLAD256 yet gives the highest accuracies of all encoding methods.

For a better understanding of the performance contribution, Table 3.5 presents the accuracy comparison of each representation component of SD-VLAD with the similar approach [57]. We report the comparison for first- and second-order statistics, and as we do not consider the third-order statistics we just report the performance of [57] in this case. We can see that our formulation for first-order statistics gives slightly better results than the approach in [57], which for their first order statistics represents the

	HOG	HOF	MBHx	MBHy	HMG	sec/video	frame/sec
VLAD256	0.717	0.812	0.789	0.802	0.809	0.20	971
VLAD512	0.735	0.823	0.801	0.816	0.823	0.32	620
H-VLAD [57]	0.755	0.825	0.800	0.809	0.820	0.38	520
SD-VLAD	**0.772**	**0.835**	**0.802**	**0.822**	**0.834**	0.23	848

Table 3.4: A closer comparison of the computational cost and the accuracy with the direct competitors on the encoding approach (results reported on the UCF50 dataset, best results are in bold).

	HOG	HOF	MBHx	MBHy	HMG
H-VLAD [57] first-order (=VLAD256)	0.717	**0.812**	0.789	0.802	0.809
SD-VLAD first-order	**0.721**	**0.812**	**0.793**	**0.805**	**0.816**
H-VLAD [57] second-order	0.724	0.809	0.774	0.782	0.786
SD-VLAD second-order	**0.760**	**0.822**	**0.785**	**0.796**	**0.810**
H-VLAD [57] third-order	*0.606*	*0.709*	*0.708*	*0.705*	*0.700*

Table 3.5: A deeper comparison: for each component. We report the performance comparison for each part of our formulation of the encoding method with the approach [57] (results reported on the UCF50 dataset, best results are in bold, in italic are represented the results for the third-order statistics of [57], as we do not use the third-order information, it is not possible a direct comparison).

	HOG	HOF	MBH	HMG	sec/video	frames/sec
IDT [88]	0.826	0.851	0.889	-	50.5	3.9
dense	0.820	0.834	0.832	0.850	10.9	18.0

Table 3.6: Comparison to IDT [88] in terms of accuracy and computational cost on the UCF50 dataset.

original VLAD formulation. This slightly improvement is mainly due to the standard deviation integration from Equation (3.5).

The main difference in performance is reflected for the second-order statistics, where SD-VLAD outperforms [57] consistently for all five descriptors. This consistent improvement for the second-order of VLAD-DV is mainly due to the consideration of the global mean of the cluster instead of the local mean of the descriptors when computing the standard deviation for the local descriptors. Remarkably that the shape difference formulation of our SD-VLAD provides only by itself a better representation than VLAD. For example, VLAD obtains for HOG an accuracy of 0.717, while the shape difference for SD-VLAD (from Equation (3.7)) obtains an accuracy of 0.760, and in this case the vector lengths are equal and the computational costs are similar. Therefore, the shape difference for SD-VLAD can replace directly VLAD in many situations. After this set of experiments we can take the conclusion that SD-VLAD is a proper method for a trade-off between the efficiency and the performance, thus, for the next experiments we will continue with FV as the choice when accuracy is crucially important and with SD-VLAD for the trade-off.

3.5.5 Comparison with Improved Dense Trajectories

The Improved Dense Trajectories (IDT) [88] represents a state-of-the-art video representation approach. We compare our descriptor extraction approach with IDT in terms of accuracy and of computational efficiency. As

in [88] where there are reported the results for FV with 256 clusters, we perform the comparison with our dense approach using the same encoding method. As the code in [88] provides four main descriptors (HOG, HOF, MBHx and MBHy), for a fair comparison we compare its extraction time with dense extraction time also for four descriptors: HOF, MBHx, MBHy and HMG. Notice that dense HOG and HMG have similar computational time, so it is not relevant for time measurement which one is selected. The comparison with IDT is presented in Table 3.6. For the computational efficiency the dense approach outperforms by a large margin IDT, being 4.6 times faster. The dense approach is able to process a video with 18 frames per second while IDT can process only 3.9 frames per second. Even though [88] provides a fast code in C++, the Matlab implementation for dense descriptors is considerably less demanding for the computational cost due to several factors. First, IDT uses a more complicate algorithm to extract the descriptors and furthermore, their approach improves the accuracy by canceling the camera motion. For doing this it is necessary to compute two times the optical flow, which makes the algorithm more demanding for computational efficiency. Another reason is that the dense descriptors are computed more efficiently, being able to reuse the blocks for many times and without the need to compute any trajectories. Very interesting is the fact that our HMG descriptor is able to compete even with HOF from IDT, with almost similar performance of 0.85.

3.5.6 Frame subsampling

Subsequent video frames contain similar information. In this set of experiments we investigate the impact on the accuracy results when frames are skipped, with the goal of speeding up the feature extraction process. We evaluate when skipping 2, 3 and 6 frames. The modality of frame subsampling is similar as in [82]. For a fair comparison, the features describe the

	$\binom{\text{frames/block}}{\text{sample rate}}$	$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
HOG	SD-VLAD	0.772	0.768	0.770	0.775
	FV	0.820	0.817	0.814	0.820
	sec/video	2.67	1.54	1.15	0.78
	frame/sec†	73.50	127.07	170.99	250.79
HOF	SD-VLAD	0.835	0.823	0.812	0.798
	FV	0.834	0.820	0.817	0.799
	sec/video	4.03	2.28	1.68	1.06
	frame/sec†	48.61	86.04	116.27	184.73
MBHx	SD-VLAD	0.802	0.793	0.792	0.778
	FV	0.816	0.806	0.797	0.779
	sec/video	4.37	2.45	1.80	1.12
	frame/sec†	44.80	80.03	108.96	174.60
MBHy	SD-VLAD	0.822	0.816	0.811	0.796
	FV	0.824	0.819	0.814	0.794
	sec/video	4.37	2.44	1.80	1.12
	frame/sec†	44.84	80.27	108.67	174.37
HMG	SD-VLAD	0.834	0.835	0.835	0.822
	FV	0.850	0.845	0.843	0.829
	sec/video	2.73	1.59	1.19	0.80
	frame/sec†	71.73	123.47	164.17	245.45

Table 3.7: Trade-off between frame sampling rate and accuracy for the UCF50 dataset. We keep video volumes from which descriptors are extracted the same for all sampling rates. †Frames/second is measured in terms of the total number of frames of the original video, not in terms of how many frames are actually processed during descriptor extraction.

	$\binom{\text{frames/block}}{\text{sample rate}}$	$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
HOG	SD-VLAD	0.653	0.658	0.662	0.664
	FV	0.708	0.719	0.721	0.722
HOF	SD-VLAD	0.740	0.725	0.719	0.697
	FV	0.741	0.729	0.719	0.700
MBHx	SD-VLAD	0.709	0.703	0.696	0.681
	FV	0.729	0.718	0.708	0.688
MBHy	SD-VLAD	0.728	0.723	0.717	0.701
	FV	0.742	0.731	0.723	0.707
HMG	SD-VLAD	0.747	0.753	0.745	0.743
	FV	0.771	0.780	0.771	0.757

Table 3.8: Trade-off between frame sampling rate and accuracy for the UCF101 dataset. We keep video volumes from which descriptors are extracted the same for all sampling rates.

	$\binom{\text{frames/block}}{\text{sample rate}}$	$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
HOG	SD-VLAD	0.367	0.380	0.378	0.380
	FV	0.406	0.399	0.390	0.399
HOF	SD-VLAD	0.433	0.420	0.411	0.392
	FV	0.433	0.424	0.408	0.388
MBHx	SD-VLAD	0.399	0.395	0.393	0.376
	FV	0.407	0.400	0.397	0.371
MBHy	SD-VLAD	0.395	0.405	0.397	0.384
	FV	0.405	0.402	0.401	0.377
HMG	SD-VLAD	0.418	0.417	0.408	0.409
	FV	0.440	0.438	0.435	0.414

Table 3.9: Trade-off between frame sampling rate and accuracy for the HMDB51 dataset. We keep video volumes from which descriptors are extracted the same for all sampling rates.

same video volume for the process of subsampling frames. For instance, if we sample every 2 frames, our baseline for the size of the block of 8 by 8 pixels by 6 frames is changing to 8 by 8 by 3 frames; for skipping 3 frames we have only 8 by 8 pixels by 2 frames; and when sampling every 6 frames the block size became 8 by 8 pixels by 1 frame.

The results for frame subsampling are presented in Table 3.7 for UCF50 dataset, in Table 3.8 for UCF101 and in Table 3.9 for HMDB51. In Table 3.7 we present the computational cost for the descriptor extraction. We reported the computational cost measurements only for UCF50 dataset in Table 3.7, as the video resolution is similar for all tree datasets, thus, the numbers reported for the measurements for the frames per second are valid also for UCF101 and HMDB51 datasets.

By subsampling frames the computational cost is significantly improved, making the pipeline more efficient. HOG descriptor is not negatively affected by skipping frames because this descriptor captures the appearance information and subsequent video frames contain similar information. Therefore, for HOG descriptor we can skip frames with a step of 6 without loosing accuracy for both FV and SD-VLAD, being able to process more than 250 frames from the video per second. For the descriptors based on optical flow a frame sampling rate of 3 gives a good trade-off, improving considerably the computational cost. HMG with SD-VLAD can have a frame sampling rate of 6 without decreasing significantly the accuracy, and a frame sampling rate of 2 almost without affecting accuracy.

3.5.7 Real-time video classification

For a real-time video classification system we propose the framework illustrated in Fig. 3.7. We use dense descriptors due to their efficiency, additionally we speed-up the pipeline by using a frame sampling rate (FSR) of 6 (thus, frames/block=1) for HOG and HMG and a FSR of 3 (thus,

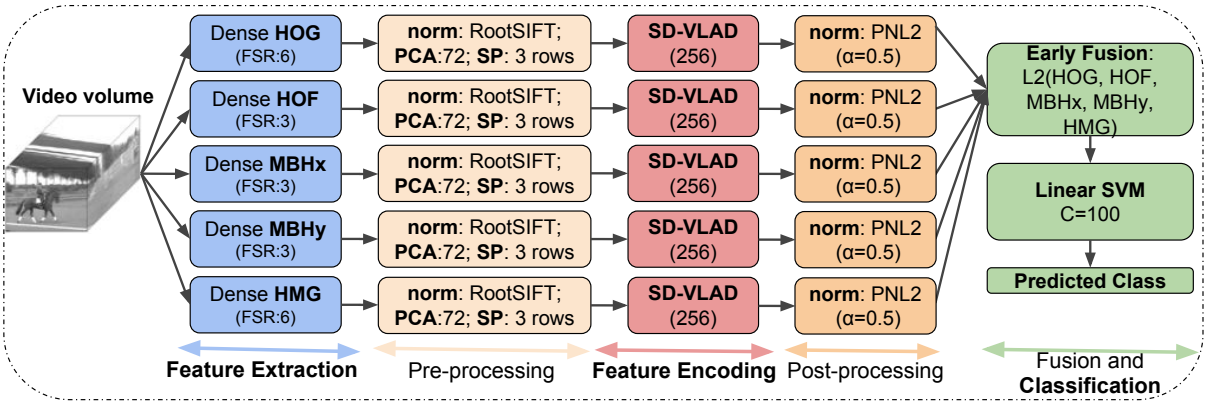

Figure 3.7: The pipeline for real-time video classification. This framework can process the video at a speed of 39 frames/second and yields an accuracy of 0.845 on UCF50 dataset, 0.767 on UCF101 and 0.470 on HMDB51.

frames/block=2) for HOF, MBHx add MBHy. It is recommended that the FSR to be equal for all descriptors based on optical flow for the reason of computing the optical flow only once and use it for all of them. After we extract the descriptors, each of them follows its separate path through the pipeline.

We initially normalize each descriptor using RootSIFT [1], then we apply PCA to reduce their dimension by a factor of 2 (from 144 dimensions to 72). Before encoding we apply a spatial pyramid representation (SP), dividing (in 3 rows) the descriptors based on their location in the frame, in bottom, middle and top part of the frame. Then for each group of the descriptors we apply our efficient encoding method, SD-VLAD, with 256 visual words. The output vector of the encoding method is normalized using PNL2 with $\alpha = 0.5$. In the end we perform an early fusion by concatenating all five descriptors, then we apply L2 normalization on the final representation for making unit length. After we compute the distances we feed them to a linear SVM ($C = 100$) to get the final result, the predicted class for a video.

The pipeline from Fig. 3.7 is able to obtain more than real-time pro-

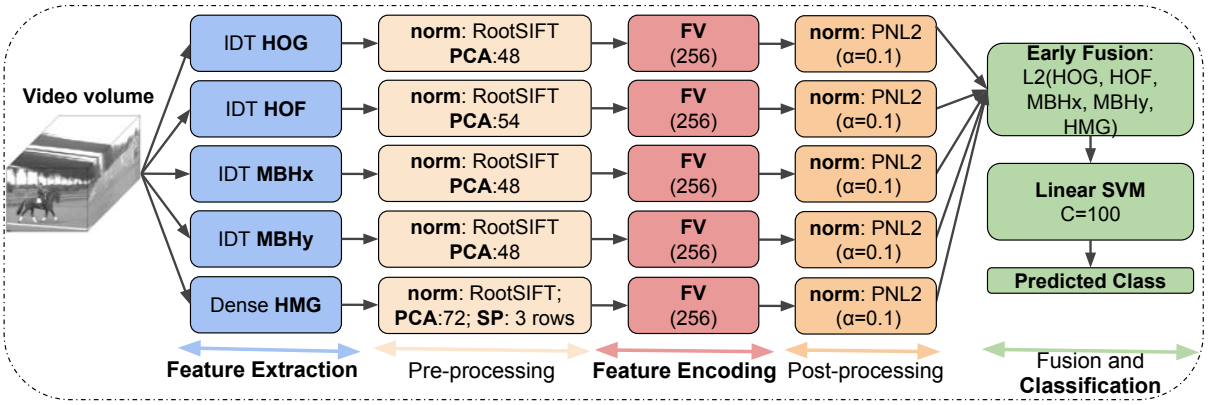

Figure 3.8: The pipeline for accurate video classification. This framework yields an accuracy of 0.930 on UCF50 dataset, 0.881 on UCF101 and 0.610 on HMDB51, but can process only 3 frames/second.

cessing rate, being capable to run with 39 frames per second and to obtain an accuracy of 0.845 for UCF50 dataset, 0.767 on UCF101 and 0.470 on HMDB51.

3.5.8 Comparison to state-of-the-art

When accuracy is crucially important for the application we recommend using Fisher Vector (with 256 clusters) for feature encoding and combining our HMG descriptor with IDT descriptors. The HMG descriptor is used in this case without skipping frames. Our pipeline for accurate action recognition can be visualized in Fig. 3.8. We extract all the descriptors of IDT (HOG, HOF, MBHx and MBHy) with the default settings provided in [88]. We perform early fusion between HMG and IDT by concatenating all features. For all features we apply separately, before early fusion, PNL2 normalization with $\alpha = 0.1$. This combination improves the accuracy from 0.912 reported in [88] (for IDT) to 0.930 for UCF50 dataset, from 0.859 [89] to 0.881 for UCF101 and from 0.572 [88] to 0.610 for HMDB51. This significant improvement in performance shows that our HMG descriptor brings complementary information for IDT and can be used to boost the perfor-

UCF50 (Acc.)		UCF101 (Acc.)		HMDB51 (Acc.)	
Kliper-Gross et al. [42]	0.727	Karpathy et al. [40]	0.654	Jain et al. [34]	0.521
Solmaz et al. [75]	0.737	Wang et al. [89]	0.859	Oneata et al. [55]	0.548
Reddy et al. [63]	0.769	Wang et al. [87]	0.860	Park et al. [56]	0.562
Uijlings et al. [81]	0.809	Peng et al. [57]	0.877	Wang et al. [88]	0.572
Uijlings et al. [82]	0.818	Peng et al. [58]	0.879	Sun et al. [79]	0.591
Wang et al. [86]	0.856	Simonyan et al. [70]	0.880	Simonyan et al. [70]	0.594
Wang et al. [88]	0.912	Sun et al. [79]	0.881	Peng et al. [57]	0.598
Wang et al. [87]	0.917	Park et al. [56]	**0.891**	Wang et al. [87]	0.601
Peng et al. [58]	0.923	Bilen et al. [6]	**0.891**	Bilen et al. [6]	**0.652**
HMG + iDT	**0.930**	HMG + iDT	0.881	HMG + iDT	0.610

Table 3.10: Comparison to the state-of-the-art.

mance of the system. However, using IDT and FV makes the pipeline more demanding for the computational cost, enabling to process only around 3 frames per second, making it not suitable for real-time applications.

Table 3.10 presents the performance comparison of our accurate pipeline from Fig. 3.8 with state-of-the-art approaches. The proposed combination for accuracy between HMG and IDT obtains state-of the art results on UCF50 and competitive results on UCF101 and HMDB51. Our framework outperforms all the methods based on hand-crafted features, including the recent work of [87], which considers as encoding method the spatial FV [43] together with spatio-temporal pyramid [46]. Our results are better than [87] which considers a hybrid representation by combining two different representations. While the approaches based on learned features (deep learning) such as [6, 56] obtain state-of-the-art results, remarkably that our pipeline is able to outperform many other well-known approaches based on deep learning such as [40, 70].

3.6 Conclusion

In this work we propose an efficient pipeline for action recognition. As two critical factors for a powerful action recognition pipeline are represented by descriptor extraction and descriptor encoding steps, we propose a new solution for both of them. We introduce in this work a new descriptor, Histograms of Motion Gradients (HMG), that captures motion information without the need of computing the optical flow, which obtains very competitive results while achieving a low computational complexity. Regarding the descriptor encoding we propose Shape Difference for VLAD (SD-VLAD). This approach captures information regarding the distribution shape of the descriptors, providing the best trade-off between computational cost and accuracy.

Based on our solutions for descriptor extraction and encoding, we propose an accurate and a real-time video classification pipeline. We test our approach on the challenging datasets: UCF50, UCF101 and HMDB51, being able to outperform well-know competitive approaches on this task.

The next chapter continues to work on feature encoding by proposing a new feature encoding approach specifically designed for deep features.

Chapter 4

Spatio-Temporal Vector of Locally Max Pooled Features for Action Recognition in Videos[1]

We introduce Spatio-Temporal Vector of Locally Max Pooled Features (ST-VLMPF), a super vector-based encoding method specifically designed for local deep features encoding. The proposed method addresses an important problem of video understanding: how to build a video representation that incorporates the CNN features over the entire video. Feature assignment is carried out at two levels, by using the similarity and spatio-temporal information. For each assignment we build a specific encoding, focused on the nature of deep features, with the goal to capture the highest feature responses from the highest neuron activation of the network. Our ST-VLMPF clearly provides a more reliable video representation than some of the most widely used and powerful encoding approaches (Improved Fisher Vectors and Vector of Locally Aggregated Descriptors), while maintaining a low computational complexity. We conduct experiments on three action recognition datasets: HMDB51, UCF50 and UCF101. Our pipeline obtains

[1]Duta, I.C.; Ionescu, B.; Aizawa, K. and Sebe, N. "Spatio-Temporal Vector of Locally Max Pooled Features for Action Recognition in Videos". In Computer Vision and Pattern Recognition (CVPR), 2017.

state-of-the-art results.

4.1 Introduction

Action recognition is still a very challenging and high computationally demanding task in computer vision, receiving a sustained attention from the research community due to its huge pool of potential applications. Its pipeline can be broken down into three main steps: feature extraction, encoding and classification. While for the classification part, the existing techniques are more mature, for feature extraction and encoding there is still a significant room for improvement. There are two main directions for feature extraction: hand-crafted and learned features (deep features). For hand-crafted category the most popular descriptors are represented by Histogram of Oriented Gradients (HOG) [15, 46], Histogram of Optical Flow (HOF) [46] and Motion Boundary Histograms (MBH) [16]. These descriptors are extracted from a video using different approaches to establish the region of extraction, such as at interest points [45], using a dense sampling [82, 90], along motion trajectories [78, 86, 88]. Recently, the features learned with a deep neural network represent a breakthrough in research, obtaining impressive results [6, 40, 56, 70, 71, 79, 80, 92, 96].

The feature encoding is one of the crucial steps for the system performance. Super vector-based encoding methods represent one of the most powerful solutions to build the final representation that servers as an input for a classifier. Improved Fisher Vectors (iFV) [61] and Vector of Locally Aggregated Descriptors (VLAD) [36] proved their superiority over other encoding methods in many works and are presented as state-of-the-art approaches for the encoding step [24, 52, 59, 81, 82, 88]. One of the shortcomings for current standard encodings is the lack of considering the spatio-temporal information, which is crucially important, especially when

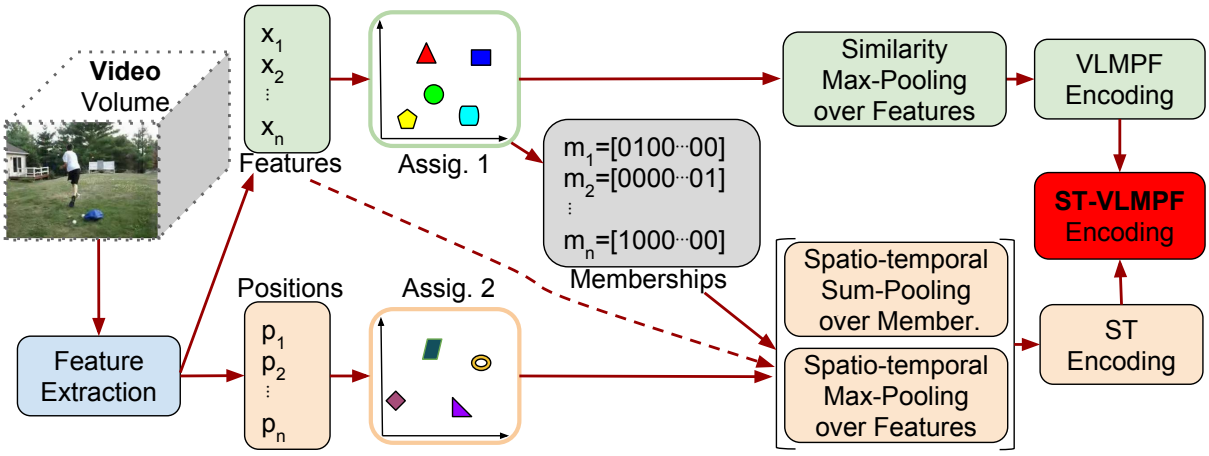

Figure 4.1: The ST-VLMPF framework for deep features encoding.

dealing with videos. These encoding approaches are built around hand-crafted features. However, a new trend is using deep features, as they obtain promising results over the traditional hand-crafted features. Many recent works apply these encoding methods also on deep features. Nevertheless, there is not yet a mature pipeline established for using these new features as their nature and behavior are implicitly different from the hand-designed category.

Deep features are learned throughout a deep neural network, providing high discriminative power on the upper layers of the network, with high level information such as objects, while hand-crafted features are manually designed and usually contain low-level information such as edges. Deep features are characterized also by their high sparsity. For instance, in [71, 92] the feature maps extracted from the upper layers of the networks (which are often used in the works as features), can contain a sparsity level of more than 90%, while for hand-crafted features, as in [82, 90], the level of sparsity is negligible. Most of the current encoding methods, such as iFV and VLAD, are built to capture high statistical information to improve the performance. In general, for hand-crafted features, iFV works better than VLAD [59, 82] due to the fact that iFV captures first- and second-order

statistics, while VLAD is based only on first order. While for hand-crafted features using more statistical information improves significantly the performance, considering the difference between them, for deep features using high-order statistics may not guarantee the performance improvement. As a matter of fact, in many recent works, as in [94], VLAD encoding is underlined as outperforming iFV, when using deep features. This aspect is also verified in our experiments, where iFV does not guarantee a better performance than VLAD. This shows that a simpler encoding method can perform better than an approach which rely on high order information. This is a completely opposite behavior in comparison with hand-crafted features. Considering all of these, we argue that a new encoding method, specifically designed for deep features, can provide better results.

With the current high availability of off-the-shelf pre-trained neural networks, many researchers use them only as feature extraction tools, as retraining or fine tuning is more demanding in many aspects. Thus, it is necessary for a performant deep features encoding approach. One of the major shortcomings of the current ConvNets-based approaches is represented by the fact that the networks take into account one or several staked frames (for instance, 10-staked optical flow fields [70, 92]). Each sampled input for the network is assigned to the overall video label from which it belongs to. The issue is that if we consider such a short number of frames as input to the network, then it may not correctly reflect the overall video label, resulting in a false label input. The current ConvNets approach individually obtains the prediction scores from all sampled inputs from a video. Then, the final prediction for a video is computed by aggregating all these prediction scores resulted from each sampled input. However, this simple aggregation cannot completely solve the aforementioned issue. This work tackles this important problem by learning a new general representation which reflects the overall video label. This approach gives to the classifier

access to the deep features extracted over the entire video.

In this work, we propose the following main contributions: (i) *provide a new encoding approach* specifically designed for working with deep features. We exploit the nature of deep features, with the goal of capturing the highest feature responses from the highest neuron activation of the network. (ii) *efficiently incorporate the spatio-temporal information within the encoding method* by taking into account the features position and specifically encode this aspect. Spatio-temporal information is crucially important when dealing with video classification. Our final proposed encoding method (illustrated in Figure 4.1), Spatio-Temporal Vector of Locally Max Pooled Features (ST-VLMPF), performs two different assignments of the features. One is based on their similarity information, the other on the spatio-temporal information. For each resulted assignment we perform a specific encoding, by performing two max-poolings and one sum-pooling of the information. (iii) *provide an action recognition scheme* to work with deep features, which can be adopted to obtain impressive results with any already trained network, without the need for re-training or fine tuning on a particular dataset. Furthermore, our framework can easily combine different information extracted from different networks. In fact, our pipeline for action recognition provides a reliable representation outperforming the previous state-of-the-art approaches, while maintaining a low complexity. We make the code for our proposed ST-VLMPF encoding publicly available (https://iduta.github.io/software.html).

The rest of the chapter is organized as following: Section 4.2 summarizes the related works. Section 4.3 introduces our encoding method. Section 4.4 presents the local deep feature extraction pipeline. The experimental evaluation is presented in Section 4.5. The conclusions are drawn in Section 4.6.

4.2 Related work

There are many works focusing on improving the feature encoding step, as the resulted final representation, which serves as input for a classifier, is a key component for the system performance. Super vector-based encoding methods are among the most powerful representation generators. Improved Fisher Vectors (iFV) [61] is one of the state-of-the-art super vector-based encoding methods which performs a soft assignment of the features and incorporates first- and second-order information. Vector of Locally Aggregated Descriptors (VLAD) [36] is a simplification of iFV capturing only first-order information and performing a hard assignment of the features. Super Vector Coding (SVC) [99] method keeps the zero-order and first-order statistics, thus SVC can be seen as a combination between Vector Quantization (VQ) [72] and VLAD.

Many recent works try to improve the aforementioned methods. The work in [57] proposes to improve VLAD by concatenating the second- and third-order statistics, and using supervised dictionary learning. The work in [52] proposes to use Random Forests in a pruned version for the trees to build the vocabulary and then additionally concatenate second-order information similar as iFV. The works in [46, 47] consider a Spatial Pyramid approach to capture the information about features location, however, the scalability is an issue for this method, as it increases considerably the size of the final representation and it is not feasible for dividing the video in more than 4 segments. The work in [2] proposes to use intra-normalization to improve VLAD performance. In [22] is proposed a double assignment for VLAD to boost the accuracy. The work in [60] uses a multi-layer nested iFV encoding to boost the performance. Different from aforementioned methods which are initially built to encode hand-crafted features, our work proposes a method specifically designed for local deep features

encoding.

Recently, encouraged by deep learning breakthroughs, many works [6, 40, 56, 70, 71, 79, 80, 92, 96] encapsulate all three main steps: feature extraction, encoding and classification, in an end-to-end framework. The work in [70] uses two streams, to capture both appearance and motion information. The works in [29, 30] are based on rank pooling for encoding; the authors in [6] extend this idea to dynamic images to create a video representation. Over the aforementioned approaches, our proposed method has the advantage of being able to use any available trained network without the need to train, re-train or fine tune it, obtaining impressive performance, even improving the original network results. Furthermore, our method can easily combine different networks, with different source of information, to create a competitive video representation.

4.3 Proposed ST-VLMPF encoding method

In this section we introduce our proposed encoding approach for deep features, Spatio-Temporal Vector of Locally Max Pooled Features (ST-VLMPF). We initially learn a codebook, C, using k-means, from a large subset of randomly selected features extracted from a subset of videos from the dataset. The outcome represents k_1 visual words, C=$\{c_1, c_2, ..., c_{k_1}\}$, which are basically the means of each feature cluster learned with k-means. When we extract the features we also retain their location within the video. For each feature we associate a position p:

$$p = (\bar{x}, \bar{y}, \bar{t});\ \bar{x} = \frac{x}{h},\ \bar{y} = \frac{y}{w},\ \bar{t} = \frac{t}{\#fr} \tag{4.1}$$

where h, w and $\#fr$ represent the height, width and the number of frames of the video. Therefore, $\bar{x}$, $\bar{y}$, $\bar{t}$ correspond to the normalized x, y, t position with respect to the video. This normalization guarantees that the position values range between the same interval [0;1] for any video.

In parallel with the first codebook C, we also learn with k-means a second codebook, PC=$\{pc_1, pc_2, ..., pc_{k_2}\}$, from the corresponding selected feature locations. The size of PC is k_2 and the outcome represents the positions codebook. The codebook PC is computed from the location information of the features used for the first codebook C. This is an automatic way to propose a k_2 spatio-temporal video divisions.

After building the codebooks, we can start creating the final video representation, which serves as input for a classifier. Figure 4.1 sketches the pipeline that a video traverses to obtain its final representation. The framework starts by extracting the local features from the video (see Section 4.4). The video is represented by its extracted local features X=$\{\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_\mathrm{n}\} \in R^{\mathrm{n}\times\mathrm{d}}$, where d is the feature dimensionality and n is the total number of the local features of the video. Together with the local features, we retain, as explained above, their positions P=$\{p_1, p_2, ..., p_n\} \in R^{\mathrm{n}\times 3}$.

Our proposed encoding method performs two hard assignments using the obtained codebooks, the first is based on the features similarity and the second is based on their positions. For the first assignment each local video feature $\mathrm{x_j}$ (j=$1, ..., n$) is assigned to its nearest visual word from the codebook C. Then, over the groups of features assigned to a cluster c_i (i=$1, ..., k_1$) we compute a vector representation v^{c_i}=$[v_1^{c_i}, v_2^{c_i}, ..., v_d^{c_i}]$, where each $v_s^{c_i}$ (s iterates over each dimension of the vector, s=$1, ..., d$) is formally computed as following:

$$v_s^{c_i} = \mathrm{sign}(\mathrm{x_{j,s}}) \max_{\mathrm{x_j:NN(x_j)}=c_i} |\mathrm{x_{j,s}}| \tag{4.2}$$

where $\mathrm{NN(x_j)}$ denotes the nearest neighborhood centroid of the codebook C for the feature $\mathrm{x_j}$, basically it guarantees that we perform separately the pooling over each group of features that are assigned to a visual word; the *sign* function returns the sign of a number and |.| represents the absolute value.

Basically, Equation 4.2 obtains the maximum absolute value while keeping the initial sign for the returned final result. In Figure 4.1 we name this similarity max-pooling over features, as the features are grouped based on their similarity and then perform max-pooling over each resulted group. The concatenation of all vectors $[v^{c_1}, v^{c_2}, ..., v^{c_{k_1}}]$, represents the VLMPF (Vector of Locally Max Pooled Features) encoding, with final vector size $(k_1 \times d)$.

After the first assignment, we also retain the centroid membership of each feature, with the objective of preserving the associated similarity-based cluster information. For each feature, we represent the membership information by a vector m with the size equal to the number of visual words k_1, where all the elements are zero, except one value (which is equal to 1) that is located on the position corresponding to the associated centroid. For instance, m=[0100...00] maps the membership feature information to the second visual word of the codebook C.

We perform a second assignment based on the features positions. The bottom part of Figure 4.1 shows this path. Each feature position p_j from P is assigned to its nearest centroid from codebook PC. After we group the features based on their location we compute another vector representation, by performing two pooling strategies: one max-pooling over the spatio-temporal clustered features and another sum-pooling over the corresponding spatio-temporal clustered features membership. We concatenate the results of these two poolings from each cluster pc_r $(r=1, ..., k_2)$. Therefore, for each spatio-temporal group of features we compute a vector representation $v^{pc_r}=[v_1^{pc_r}, v_2^{pc_r}, ..., v_d^{pc_r}]$, where each $v_s^{pc_r}$ is formally computed

as following:

$$v_s^{pc_r} = cat\Big[\ \mathrm{sign}(\mathrm{x_{j,s}}) \max_{p_j:NN(p_j)=pc_r} |\mathrm{x_{j,s}}|, \Big(\sum_{p_j:NN(p_j)=pc_r} \mathrm{m_{j,i}}\Big)^{\alpha}\ \Big] \tag{4.3}$$

where *cat* denotes the concatenation and $NN(p_j)$ denotes the nearest neighborhood visual word of the codebook PC for the feature position p_j. Due to the fact that the sum-pooling over the membership information can create peaks within the vector we normalize the result of sum-pooling similar to power normalization, with standard α=0.5. Basically, in this case we perform square root over the result of the sum-pooling to reduce the peaks within the final vector.

Differently from Equation 4.2, in Equation 4.3 we group the features based on the spatio-temporal information and then we compute the maximum absolute value while keeping the original sign over the features. We also concatenate in Equation 4.3 the membership information regarding the feature similarity obtained from the first assignment with the goal of encapsulating together with spatio-temporal information also the similarity membership of the spatio-temporal grouped features. We concatenate all these vectors $[v^{pc_1}, v^{pc_2}, ..., v^{pc_{k_2}}]$ to create the ST (Spatio-Temporal) encoding, which results in a $(k_2 \times d + k_2 \times k_1)$ vector size.

Finally, we concatenate the ST and VLMPF encodings to create the final ST-VLMPF representation, which serves as input the classifier. Therefore, the final size of the vector for ST-VLMPF representation is $(k_1 \times d) + (k_2 \times d + k_2 \times k_1)$. The goal of ST-VLMPF is to provide a reliable representation which incorporates the deep features over the entire video, providing to the classifier a more complete information for taking the right decision.

4.4 Local deep features extraction

This section presents the pipeline for local deep features extraction. The approaches based on convolutional networks (ConvNets) [6, 40, 56, 70, 71, 79, 80, 92, 96] have recently obtained very competitive results over traditional hand-crafted features. The videos contain two main sources of information: appearance and motion. In our pipeline for feature extraction we individually use three streams: a spatial stream for capturing the appearance, a temporal stream for capturing the motion and a spatio-temporal stream for capturing at the same time both appearance and motion information. The pipeline for local deep feature extraction is illustrated in Figure 4.2, where for a given video we extract, independently for each of three networks, the features maps with spatial information.

For capturing the appearance information in our spatial stream we use the VGG ConvNet in [71], which is a network with 19 layers. The local deep feature extraction pipeline for this network is depicted in the upper part of Figure 4.2. The input of VGG19 ConvNet is an image with 224×224 resolution and three channels for the color information. After we extract the individual frames from a video, we accordingly resize them to the required input size of the network. For each individual frame we take the output of the last convolutional layer with spatial information, pool5. Our choice for the convolutional layer is motivated by the fact that the deeper layers provide high discriminative information. By taking a layer with spatial information we can extract local deep features for each frame of the video, containing also the details about spatial membership of the features. The output of pool5 is a feature map with a spatial size of 7×7 and 512 channels. For extracting local deep features from a feature map we individually take each spatial location and concatenate the values along all 512 channels, obtaining local deep features with 512 dimensions. Thus,

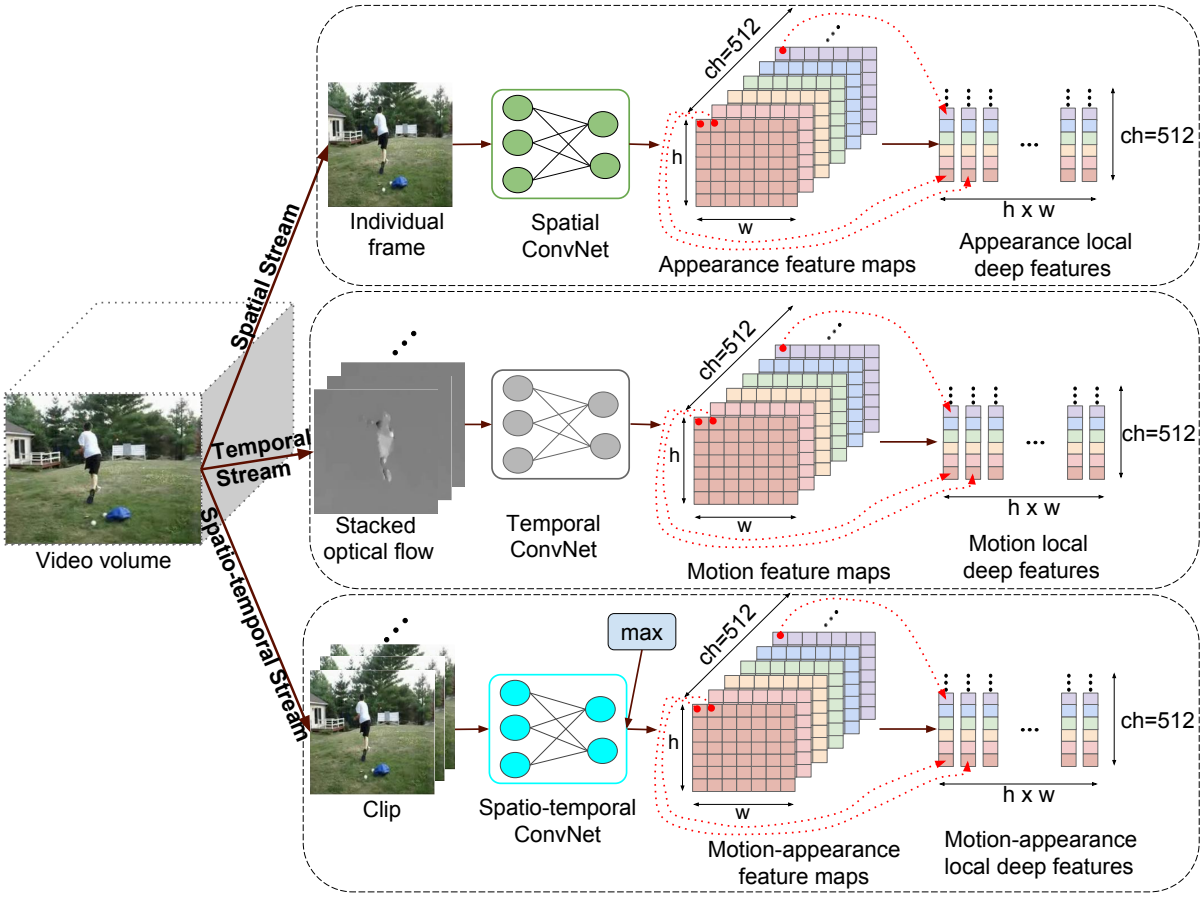

Figure 4.2: The framework for deep local feature extraction pipeline.

from each frame we obtain 7×7=49 local deep features and each feature is a 512 dimensional vector. Therefore, for each video we obtain in total $\#frames \times 49$ local deep features. SCN refers to the features extracted with this Spatial Convolutional Network.

For the motion information we use the re-trained network in [92]. This deep network, also VGG, is initially proposed in [71] and contains 16 layers. The authors in [92] re-trained the VGG ConvNet for a new task with new input data using several good practices for the network re-training, such as pre-training to initialize the network, smaller learning rate, more data augmentation techniques and high dropout ratio. The VGG ConvNet is re-trained for action recognition task using the UCF101 dataset [76]. The

input to the temporal ConvNet is 10-stacked optical flow fields, each of them with one image for vertical and one image for horizontal motion. Therefore, in total there are 20-staked optical flow images as one input to the network. To extract optical flow fields we use the OpenCV implementation of the TVL1 algorithm [97]. For the temporal ConvNet we also take the output of the last convolutional layer with structure information (pool5). The pool5 layer has the spatial size of feature maps of 7×7 and 512 channels. The final local deep features for an input are obtained by concatenating the values from each spatial location along all the channels, resulting in 49 local features for an input. This results in $(\#frames-9)\times 49$ local deep features for a video using the temporal ConvNet. TCN refers to the features extracted with this Temporal Convolutional Network.

For the spatio-temporal stream, represented at the bottom part of Figure 4.2, we use the 3D ConvNet [80]. This network is trained on Sports-1M dataset [40] and contains 16 layers. The network is designed to capture both appearance and motion information by using 3D convolutional kernels. The input of the network is a 16 frame-long clip extracted from the video. Similar to the previous two networks used in our pipeline, we use a sampling step size of one frame to iterate over the frames of the video for creating the input clips. As the last layer of this network with spatial information has the size of the feature maps of only 4×4, we consider in our pipeline one layer before, which is called conv5b. The conv5b layer has a similar spatial size of the feature maps as the previous two networks i.e., 7×7 and similar number of channels i.e., 512. However, the conv5b layer contains two features maps, each of them 7×7×512. In our pipeline, for this network, for an input, we build only one feature map of 7×7×512 by taking the maximum value for each position of the both feature maps from conv5b. Then, we can extract the local deep feature similar to the previous two networks. For the 3D network, the total number of local

deep features is $(\#frames-15)\times7\times7$ for an input video. Each resulted local deep feature is a vector with also 512 dimensions. We refer to the features extracted with this Convolutional 3D network as C3D. For all resulted local deep features from these three networks, the normalized positions, needed for ST-VLMPF, are extracted based on the localization on the feature maps.

4.5 Experimental Evaluation

This section presents the experimental part, where we test our proposed framework in the context of action recognition.

4.5.1 Datasets

We evaluate our framework on three of the most popular and challenging datasets for action recognition: HMDB51 [44], UCF50 [63], and UCF101 [76].

The HMDB51 dataset [44] contains 51 action categories, with a total of 6,766 video clips. We use the original non-stabilized videos, and we follow the original protocol using three train-test splits [44]. We report average accuracy over the three splits as performance measure.

The UCF50 dataset [63] contains 6,618 realistic videos taken from YouTube. There are 50 human action categories mutually exclusive and the videos are split into 25 predefined groups. We follow the recommended standard procedure and perform leave-one-group-out cross validation and report average classification accuracy over all 25 folds.

The UCF101 dataset [76] is a widely adopted benchmark for action recognition, consisting in 13,320 realistic videos and 101 action classes. We follow for evaluation the recommended default three training/testing splits and report the average recognition accuracy over these three splits.

4.5.2 Experimental setup

For the motion stream of the local deep feature extraction pipeline, the work in [92] provides three trained models for each split of the UCF101 dataset. We accordingly use the models for each split of the UCF101 for feature extraction. For the other two datasets, HMDB51 and UCF50, we use only the model trained on the split1 of UCF101 to extract the local deep features.

We compare our proposed ST-VLMPF encoding method with two state-of-the-art approaches for feature encoding: improved Fisher Vectors (iFV) [61] and Vector of Locally Aggregated Descriptors (VLAD) [36]. We create the codebooks from 500K random selected features extracted from a subset of videos. We set the size of the codebook to 256 visual words, which is the standard adopted size, widely used by the community when using super vector-based encoding methods. Setting also the size of codebook C (k_1=256) for ST-VLMPF the same as for the other encoding methods makes easier to compare them and also it is a fair comparison having a similar number of visual words for all super vector-based encoding methods.

When using our encoding method, ST-VLMPF, we L2 normalize the final video representation vector before classification. Many works, such as [82, 88], indicate that iFV and VLAD perform better if after feature encoding the Power Normalization (PN) is applied followed by L2-normalization ($||sign(\mathrm{x})|\mathrm{x}|^{\alpha}||$). We follow this line for iFV and VLAD, setting α to the standard widely used value of 0.5. The reason for which iFV and VLAD work better when using PN is due to the fact that their resulted final representation contains large peaks within the vector and PN helps to reduce them and make the vector smoother. Instead, in our application, ST-VLMPF does not provide a final vector containing large peaks, therefore, it is not necessary to apply also PN. For the classification part, in all the

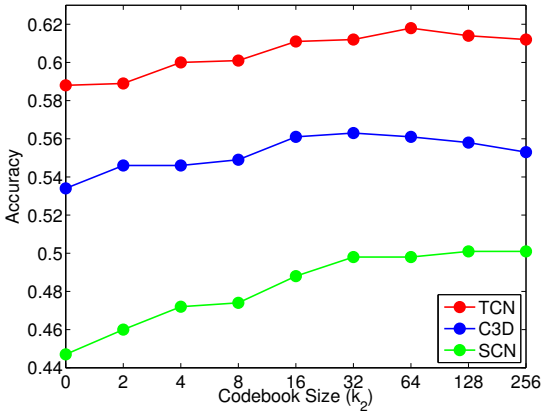

Figure 4.3: Evaluation of the spatio-temporal divisions of the video.

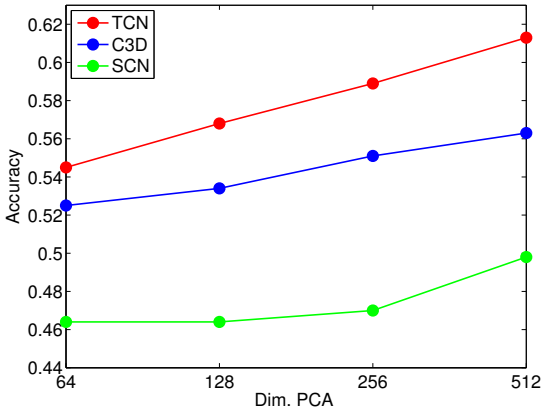

Figure 4.4: Evaluation of the dimensionality reduction with PCA.

	SCN		TCN		C3D	
	256	512	256	512	256	512
VLMPF	43.5	44.7	56.6	58.8	52.8	53.4
ST-VLMPF	**47.0**	**49.8**	**58.9**	**61.3**	**55.1**	**56.3**

Table 4.1: The final accuracy on HMDB51 using 32 spatio-temporal video divisions, with 256 and 512 feature dimensionality. We report also the results when the spatio-temporal information is not used (VLMPF).

experiments we use a linear one-vs-all SVM with the parameter C=100.

4.5.3 Parameter tuning

We present the parameter tuning regarding the number of divisions of a video and the features dimensionality. All the tuning experiments are reported on the HMDB51 dataset.

Figure 4.3 presents the evaluation of parameter k_2, which denotes the size of codebook PC. The k_2 parameter represents the number of video divisions used for our ST-VLMPF encoding approach. We report the evaluation on all three local deep features considered: SCN, TCN and C3D; keeping all 512 dimensions of the original local deep features. The 0 value

illustrated in Figure 4.3 represents the case when the spatio-temporal information is not considered, which refers to the VLMPF encoding from Figure 4.1. Remarkably, the performance of ST-VLMPF for all three features has a continuous significant boost in accuracy when increasing the video divisions, until k_2=32. This graph clearly shows that our approach to incorporate spatio-tempral information within the encoding process brings significant gain on the final accuracy for an action recognition system. While for the C3D features the increase in the accuracy stops around the value of k_2=32, for the SCN and TCN the accuracy still continue to have a slight increase. However, we set k_2=32 for our ST-VLMPF encoding in all remaining experiments in this work, as this value provides a good trade-off between accuracy and computational cost and the size of the final video representation.

Figure 4.4 illustrates the evaluation when using PCA to reduce the feature dimensionality and decorrelate the data. From the graph we can see that for all features the accuracy is drastically influenced by the number of dimensions kept. Decreasing the features dimensionality from the original size of 512 to 64 causes a considerable drop in accuracy for SCN from 0.498 to 0.464, for TCN from 0.613 to 0.545 and for C3D from 0.563 to 0.525. For the next experiments we will consider the original features dimensionality of 512 and also when the dimensionality is decreased to 256.

Table 4.1 summarizes the performance numbers obtained for all three features. This table includes the results with two settings for feature dimensionality: 256 and 512. We report also the results when the spatio-temporal information is not used within the encoding process (VLMPF). In this way we can directly observe the benefit of incorporating the spatio-temporal information in the encoding method over for the performance of the system.

	HMDB51 (%)						UCF50 (%)						UCF101 (%)					
	SCN		TCN		C3D		SCN		TCN		C3D		SCN		TCN		C3D	
	256	512	256	512	256	512	256	512	256	512	256	512	256	512	256	512	256	512
iFV	36.6	41.8	51.0	56.6	46.1	49.0	75.7	81.0	95.2	96.1	84.7	88.8	67.8	74.1	84.1	85.4	77.7	79.8
VLAD	37.2	40.3	51.1	53.9	46.8	49.1	78.4	80.2	95.5	95.4	86.4	89.0	69.9	73.4	83.7	85.2	78.6	81.4
ST-VLMPF	**47.0**	**49.8**	**58.9**	**61.3**	**55.1**	**56.3**	**86.3**	**87.7**	**97.1**	**97.2**	**94.1**	**94.7**	**80.4**	**81.8**	**86.6**	**87.3**	**85.5**	**86.2**

Table 4.2: Accuracy comparison on all three datasets. Best results are in bold.

4.5.4 Comparison to other encoding approaches

In this part we present the comparison of our ST-VLMPF encoding method, with VLAD and iFV, in terms of accuracy and computational efficiency.

Accuracy comparison. We present the comparison of ST-VLMPF with VLAD and iFV in terms of accuracy over three datasets: HMDB51, UCF50 and UCF101. We report the comparison results with the features dimensionality of 256 and the 512. Table 4.2 shows the comparison accuracy results for all three datasets. On the challenging HMDB51 dataset ST-VLMPF clearly outperforms by a large margin iFV and VLAD for all three features. For instance, for SCN with 256 dimensionality, ST-VLMPF obtains with 9.8 percentage points more than VLAD and with 10.4 percentage points more than iFV. Similar results are reported for UCF50 and UCF101 respectively, where we can see that our proposed encoding method, ST-VLMPF, outperforms also by a large margin iFV and VLAD in all the cases, showing the effectiveness of our representation. We can also see from Table 4.1 that our method without spatio-temporal information, still outperforms iFV and VLAD.

Efficiency comparison. Table 4.3 presents an efficiency comparison of our ST-VLMPF with iFV and VLAD. The timing measurements are performed on a single core Intel(R) Xeon(R) CPU E5-2690 2.60GHz, using 500 randomly sampled videos from HMDB51 dataset.

We report the average number of frames per second and number of sec-

	SCN 256		SCN 512		TCN 256		TCN 512		C3D 256		C3D 512		256	512
	fr/sec	sec/vid	fr/sec	sec/vid	fr/sec	sec/vid	fr/sec	sec/vid	fr/sec	sec/vid	fr/sec	sec/vid	dim	dim
iFV	253.2	0.357	168.7	0.536	301.4	0.300	197.6	0.457	308.7	0.293	202.3	0.447	131,072	262,144
VLAD	1967.5	0.046	1143.8	0.079	2213.8	0.041	1299.5	0.070	2372.5	0.038	1375.0	0.066	**65,536**	**131,072**
VLMPF	**2049.4**	**0.044**	**1192.6**	**0.076**	**2329.2**	**0.039**	**1370.9**	**0.066**	**2455.0**	**0.037**	**1426.0**	**0.063**	**65,536**	**131,072**
ST-VLMPF	1531.1	0.059	964.7	0.094	1741.0	0.052	1062.0	0.085	1769.6	0.051	1086.5	0.083	81,920	155,648

Table 4.3: Computational efficiency comparison. We report the number of frames per second (fr/sec) and seconds per video (sec/vid). Last two columns show the dimensionality generated by each encoding method for 256 and 512 feature dimensionality. Best results are in bold.

onds per video that an encoding method can process for creating a video representation. For our encoding method we report also the results without using the spatio-temporal information (VLMPF) for directly observing the cost of adding the spatio-temporal encoding. We can see that by far the most expensive method for the computational cost is iFV. This is due to the fact that the method uses soft assignment and high order statistics to create the final representation. The VLAD encoding is slightly slower than VLMPF and this is due to the computation of the residuals. The computational cost for our ST-VLMPF is comparable with VLAD, however, it is more efficient than iFV, being more than 5 times faster.

The last two columns of Table 4.3 present the dimensionality of the generated video representations for each encoding method. We can see that iFV is more demanding, generating a large dimensionality, while ST-VLMPF is comparable to VLAD. Even though the generated dimensionality is relatively high, in the case of a linear SVM (as we use in this work) for ST-VLMPF with 512 feature dimensionality, the classification time to get the predicted class for a given video representation is less than 0.001 seconds, therefore, this is a negligible cost.

	HMDB51 (%)			UCF50* (%)			UCF101 (%)		
	DF	DF+HMG	DF+HMG+iDT	DF	DF+HMG	DF+HMG+iDT	DF	DF+HMG	DF+HMG+iDT
Early	68.6	69.5	71.7	95.0	95.3	96.7	93.5	**94.0**	**94.3**
sLate	66.4	66.5	68.8	94.2	94.4	95.6	92.0	92.5	92.4
wLate	67.6	67.8	70.9	94.8	95.1	96.6	92.2	92.7	93.4
sDouble	68.3	68.4	70.3	94.6	94.9	96.1	92.6	93.1	92.8
wDouble	**69.5**	**70.3**	**73.1**	**95.1**	**95.4**	**97.0**	**93.6**	**94.0**	**94.3**

Table 4.4: Fusion strategies. DF (Deep Features) represent all three local deep features (SCN, TCN, C3D), HMG (Histograms of Motion Gradients) [23] and iDT (improved Dense Trajectories) [88] is represented with HOG, HOF, MBHx and MBHy. The best performance results are in bold for each fusion type over each feature representation combination. The best result over each dataset is also underlined. (*TCN features are not considered for UCF50 dataset as explained above.)

4.5.5 Fusion strategies

The previous results show that our ST-VLMPF approach obtains the best accuracy on all datasets and for all feature types. Also we show that the accuracy drops significantly when the features dimensionality decreases, therefore, to obtain the final score we use all 512 feature dimensions. Combining deep features with hand-crafted features can boost the performance of the system. Therefore, in this work we report three feature combinations: DF, DF+HMG and DF+HMG+iDT. DF (Deep Features) is represented by SCN, TCN and C3D, all deep features are encoded with our ST-VLMPF method. As previously pointed, to extract the TCN features we use a ConvNet, which is trained on the split1 of UCF101. As the UCF101 is an extension of the UCF50 dataset, to avoid the risk of overfitting, for any further fusion and for the comparison with the state-of-the-art, we excluded TCN features for the UCF50 dataset results. HMG (Histograms of Motion Gradients) [23] is a hand-crafted descriptor which efficiently captures motion information. We used the code provided by the authors with default settings for descriptor extraction, and we encode

the descriptors accordingly as recommended in the paper, using iFV. iDT (improved Dense Trajectories) [88] is a state-of-the-art hand-crafted approach, and is represented in our work by four individual hand-crafted descriptors (HOG, HOF, MBHx, MBHy). We also use the authors provided code to extract the descriptors with default settings, and create the final representation as recommended also using iFV. For all hand-crafted features we individually apply before classification PN (α=0.1) and then L2 as recommended in [23]. For these four feature combinations we evaluate different fusion strategies: Early, where after we individually build the final representation for each feature type and normalize it accordingly, we concatenate all resulted representations in a final vector, we apply L2 normalization for making unit length and then perform the classification part; sLate, where we make late fusion by making sum between the classifiers output from each representation; wLate, where we give different weights for each feature representation classifier output, and then we perform the sum. The weight combinations are tuned by taking values between 0 and 1 with the step 0.05; sDouble, where besides summing the classifier output from the individual feature representations, we also add the classifier output resulted from the early fusion; wDouble, where we tune the weight combinations for the sum, similar to wLate.

Table 4.4 shows that early fusion performs better than late fusion. Double fusion combines the benefit of both, early and late fusion, and boosts further the accuracy. For more challenging datasets such as HMDB51, combining deep features with hand-crafted features improves considerably the accuracy, while for less challenging datasets such as UCF50, the hand-crafted features do not bring significant contribution over deep features. With this framework, we obtain outstanding final results of 73.1% on HMDB51, 97.0% on UCF50 and 94.3% on UCF101.

HMDB51 (%)		UCF50* (%)		UCF101(%)	
Jain et al. [34] (2013)	52.1	Solmaz et al. [75] (2013)	73.7	Wang et al. [89] (2013)	85.9
Zhu et al. [100] (2013)	54.0	Reddy et al. [63] (2013)	76.9	Karpathy et al. [40] (2014)	65.4
Oneata et al. [55] (2013)	54.8	Shi et al. [69] (2013)	83.3	Simonyan et al. [70] (2014)	88.0
Wang et al. [88] (2013)	57.2	Wang et al. [86] (2013)	85.6	Wang et al. [87] (2015)	86.0
Kantorov et al. [38] (2014)	46.7	Wang et al. [88] (2013)	91.2	Sun et al. [79] (2015)	88.1
Simonyan et al. [70] (2014)	59.4	Ballas et al. [4] (2013)	92.8	Ng et al. [96] (2015)	88.6
Peng et al. [60] (2014)	66.8	Everts et al. [27] (2014)	72.9	Tran et al. [80] (2015)	90.4
Sun et al. [79] (2015)	59.1	Uijlings et al. [81] (2014)	80.9	Wang at al. [92] (2015)	91.4
Wang et al. [87] (2015)	60.1	Kantorov et al. [38] (2014)	82.2	Wang et al. [91] (2015)	91.5
Wang et al. [91] (2015)	65.9	Ciptadi et al. [13] (2014)	90.5	Zhang et al. [98] (2016)	86.4
Park et al. [56] (2016)	56.2	Narayan et al. [54] (2014)	92.5	Peng et al. [59] (2016)	87.9
Seo et al. [68] (2016)	58.9	Uijlings et al. [82] (2015)	81.8	Park et al [56] (2016)	89.1
Peng et al. [59] (2016)	61.1	Wang et al. [87] (2015)	91.7	Bilen et al. [6] (2016)	89.1
Yang et al. [95] (2016)	61.8	Peng et al. [59] (2016)	92.3	Diba et al. [18] (2016)	90.2
Bilen et al. [6] (2016)	65.2	Duta et al. [23] (2016)	93.0	Fernando et al. [29] (2016)	91.4
Fernando et al. [29] (2016)	66.9	Seo et al. [68] (2016)	93.7	Yang et al. [95] (2016)	91.6
Our ST-VLMPF(DF)	**69.5**	Our ST-VLMPF(DF)	**95.1**	Our ST-VLMPF(DF)	**93.6**
Our best	**73.1**	Our best	**97.0**	Our best	**94.3**

Table 4.5: Comparison to the state-of-the-art. Our ST-VLMPF(DF) represents the results obtained with only our representation over deep features (SCN, TCN and C3D). Our best is the final result from the best combination of our ST-VLMPF with hand-crafted features HMG [23] and iDT (HOG, HOF, MBHx, MBHy) [88]. (*TCN features are not considered for UCF50 dataset as explained above.)

4.5.6 Comparison to state-of-the-art

Table 4.5 presents the comparison of our final results with the state-of-the-art approaches on HMDB51, UCF50 and UCF101. For this comparison we report two final results. First result, represents only our ST-VLMPF(DF), which is obtained by using our proposed encoding method over all three deep features (SCN, TCN and C3D). The second one, is our best result reported in this work, obtained using ST-VLMPF(DF) + HMG + iDT.

Our ST-VLMPF representation outperforms the state-of-the-art ap-

proaches by a large margin on all three datasets, which demonstrates that our method provides a powerful video representation with very competitive results. Furthermore, with our best results, which use also hand-crafted features, we improve the state-of-the-art by 6.2 percentage points on the challenging HMDB51 dataset, by 3.3 percentage points on UCF50 and by 2.7 percentage points on UCF101. It is important to highlight that these results are obtained using pre-trained networks which are not re-trained or fine-tuned on our particular datasets (except TCN features for UCF101 dataset). For instance, for HMDB51 dataset all three networks did not see any training example from this dataset, and we still obtain impressive results. Therefore, our approach is also suitable in various practical scenarios when re-training or fine-tuning is more difficult to accomplish.

4.6 Conclusion

In this work we introduced the Spatio-Temporal Vector of Locally Max Pooled Features (ST-VLMPF), a super vector-based encoding method specifically designed for encoding local deep features. We also efficiently incorporate the spatio-temporal information within the encoding method, providing a significant boost in accuracy. ST-VLMPF outperforms two of the most powerful encoding methods by a large margin (Improved Fisher Vectors and Vector of Locally Aggregated Descriptors), while maintaining a low computational complexity. Our approach provides a solution for incorporating deep features over the entire video, helping to solve the issue with the false label assigned to the network input. The comparison of our action recognition pipeline with the state-of-the-art approaches over three challenging datasets proves the superiority and robustness of our video representation.

Chapter 5

Conclusions and Future Work

In this thesis, we tackled the video classification task, which is the core component of countless practical applications such as video indexing, human-computer interaction, sports analytics, elderly-care, healthcare. Two key steps for the performance of the video classification pipeline are represented by the feature extraction and encoding. This thesis focussed on providing solutions to improve these two steps.

One of the shortcomings of the descriptor extraction approaches is represented by the efficiency, which in many cases is extremely demanding for the computational cost. In Chapter 2 we proposed several solutions to speed-up the descriptor extraction step and investigated different trade-offs between accuracy and computational efficiency. We concluded the work by providing a video classification framework that is able to run in real-time frame rate.

In Chapter 3 we continued the work on descriptor extraction by providing a very efficient descriptor, Histograms of Motion Gradients (HMG), which is able to extract motion information in an efficient way, being able to run in real-time frame rate. This chapter made also the transition from descriptor extraction to another key component for video classification: feature encoding. We proposed to improve the popular approach VLAD

(Vector of Locally Aggregated Descriptors) by incorporating shape information. The results supported the benefits of the proposed approach. We concluded also this work by incorporating the proposed approaches in a real-time video classification framework.

In Chapter 4 we continued the work on feature encoding by proposing a new approach for local deep feature encoding. With this transition from hand-crafted features to deep features, it is necessary to design new encoding approaches that are effective and efficient to work with deep features. In this chapter we proposed a specifically designed encoding approach for deep features, which provided better performance in comparison with the traditional approaches when applied on deep features.

While there has been an important progress in video classification in the last several years, there is still a big room for improvement. In our vision, video understanding is much more than video classification. In fact, video classification can be considered just a fragment of video understanding. For a meaningful video understanding concept we need to combine different fields of research, such as machine learning, computer vision, natural language processing, multimedia. For instance, over a video we need to run different algorithms, such as object detection/recognition, object tracking, text/speech recognition, face detection/recognition, body pose estimation, action detection/recognition, etc. Then all this information should be combined to provide a deep understanding of the video. As an example, the user can make a query over his personal video library: "return the video where my brother dressed in a blue t-shirt is biking for a while, then he falls down". For this query we need to combine most of the aforementioned tasks to be able to extract such information. In summary, we just scratched the surface of what means video understanding, there are many aspects which request improvements, therefore, this is going to be a very important research direction.

Our future research direction will be focused on the following points:

- With this fulminant growth in multimedia content, we will focus on building scalable algorithms to combat this information explosion.
- Most of the current contributions in the community are made on top of supervised learning algorithms. In general it is very costly to get labeled data to train these algorithms. In particular for video it is even more demanding to label big datasets. In our future research we will focus to develop approaches which can extract meaningful information in an unsupervised fashion.
- Integration of knowledge from multiple sources (video, images, audio, text) to build real-life applications.
- Specifically for video classification we will focus to push forward the deep learning approaches to reduce the gap with image classification. In particular, we will also focus on building very efficient and effective 3D convolutional neural networks, which make it possible to avoid the computation of the optical flow for extracting motion information (which is a bottleneck in the video classification pipeline). These 3D convolutional neural networks are able to capture both appearance and motion information. Our future work will also focus on approaches for untrimmed videos and more complex events.

Acknowledgements

This thesis could not be accomplished without the help of many important persons. Firstly, I would like to express my sincere gratitude to my advisor, Prof. Nicu Sebe, for his excellent guidance during my PhD, I am very thankful for his support. I thank my co-adivisor, Prof. Bogdan Ionescu, he supported me during the thesis, he is also the person that guided my first steps towards Trento, I am very grateful for his help. Dr. Jasper Uijlings is the person that guided my first steps in Computer Vision field, I am very thankful for his support during my PhD. I would like also to thank my good friends, Bogdan Dragomir and George Burcea, they truly encouraged and helped me towards a career in Computer Science.

I was very lucky to have the opportunity to complete three internships during my PhD. I thank Prof. Alexander Hauptmann and Dr. Zhigang Ma for the opportunity to have an internship at Carnegie Mellon University. Prof. Kiyoharu Aizawa provided me an excellent support during my internship at the University of Tokyo, I am grateful for his help. I am thankful for the precious advice of my manager, Jayan Eledath, during my internship with the Amazon Go team.

Lastly, I would like to thank my family and friends for their continuous support. Special thanks to my wife, Alesia, for her encouragement and help, she was the first reviewer of my papers during PhD. I truly appreciate all the aforementioned people and I wish them happiness, health and bright future.

Bibliography

[1] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[2] Relja Arandjelovic and Andrew Zisserman. All about VLAD. In *CVPR*, 2013.

[3] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1):1–31, 2011.

[4] Nicolas Ballas, Yi Yang, Zhen-Zhong Lan, Bertrand Delezoide, Françoise Prêteux, and Alexander Hauptmann. Space-time robust representation for action recognition. In *ICCV*, 2013.

[5] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *CVIU*, 110(3):346–359, 2008.

[6] Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic image networks for action recognition. In *CVPR*, 2016.

[7] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[8] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*. 2004.

[9] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *TPAMI*, 33(3):500–513, 2011.

[10] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, pages 611–625, 2012.

[11] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[12] Ken Chatfield, Victor S Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.

[13] Arridhana Ciptadi, Matthew S Goodwin, and James M Rehg. Movement pattern histogram for action recognition and retrieval. In *ECCV*, 2014.

[14] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2, 2004.

[15] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[16] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*. 2006.

[17] César Roberto de Souza, Adrien Gaidon, Eleonora Vig, and Antonio Manuel López. Sympathy for the details: Dense trajectories and

hybrid classification architectures for action recognition. In *ECCV*, 2016.

[18] Ali Diba, Ali Mohammad Pazandeh, and Luc Van Gool. Efficient two-stream motion and appearance 3d cnns for video classification. In *ECCV ws*, 2016.

[19] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 65–72, 2005.

[20] Ionut C Duta, Bogdan Ionescu, Kiyoharu Aizawa, and Nicu Sebe. Simple, efficient and effective encodings of local deep features for video action recognition. In *ICMR*, 2017.

[21] Ionut C. Duta, Bogdan Ionescu, Kiyoharu Aizawa, and Nicu Sebe. Spatio-Temporal VLAD encoding for human action recognition in videos. In *MMM*, 2017.

[22] Ionut C. Duta, Tuan A. Nguyen, Kiyoharu Aizawa, Bogdan Ionescu, and Nicu Sebe. Boosting VLAD with double assignment using deep features for action recognition in videos. In *ICPR*, 2016.

[23] Ionut C. Duta, Jasper R. R. Uijlings, Tuan A Nguyen, Kiyoharu Aizawa, Alexander G Hauptmann, Bogdan Ionescu, and Nicu Sebe. Histograms of motion gradients for real-time video classification. In *CBMI*, 2016.

[24] Ionut C Duta, Jasper RR Uijlings, Bogdan Ionescu, Kiyoharu Aizawa, Alexander G Hauptmann, and Nicu Sebe. Efficient human action recognition using histograms of motion gradients and vlad

with descriptor shape information. *Multimedia Tools and Applications (MTAP)*, pages 1–28, 2017.

[25] Ionut Cosmin Duta, Bogdan Ionescu, Kiyoharu Aizawa, and Nicu Sebe. Spatio-Temporal vector of locally max pooled features for action recognition in videos. In *CVPR*, 2017.

[26] Ivo Everts, Jan C Van Gemert, and Theo Gevers. Evaluation of color STIPs for human action recognition. In *CVPR*, 2013.

[27] Ivo Everts, Jan C Van Gemert, and Theo Gevers. Evaluation of color spatio-temporal interest points for human action recognition. *TIP*, 2014.

[28] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image analysis*, pages 363–370. 2003.

[29] Basura Fernando, Peter Anderson, Marcus Hutter, and Stephen Gould. Discriminative hierarchical rank pooling for activity recognition. In *CVPR*, 2016.

[30] Basura Fernando, Efstratios Gavves, José Oramas, Amir Ghodrati, and Tinne Tuytelaars. Rank pooling for action recognition. *TPAMI*, 2016.

[31] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[32] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981.

[33] Tommi Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999.

[34] Mihir Jain, Hervé Jégou, and Patrick Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013.

[35] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[36] Hervé Jégou, Florent Perronnin, Matthijs Douze, Javier Sanchez, Pablo Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *TPAMI*, 34(9):1704–1716, 2012.

[37] Frederic Jurie and Bill Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, 2005.

[38] Vadim Kantorov and Ivan Laptev. Efficient feature extraction, encoding and classification for action recognition. In *CVPR*, 2014.

[39] Svebor Karaman, Lorenzo Seidenari, Andrew D Bagdanov, and Alberto Del Bimbo. L1-regularized logistic regression stacking and transductive CRF smoothing for action recognition in video. In *ICCV workshop on action recognition with a large number of classes*, 2013.

[40] Andrej Karpathy, George Toderici, Sachin Shetty, Tommy Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[41] Alexander Klaser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008.

[42] Orit Kliper-Gross, Yaron Gurovich, Tal Hassner, and Lior Wolf. Motion interchange patterns for action recognition in unconstrained videos. In *ECCV*, 2012.

[43] Josip Krapac, Jakob Verbeek, and Frédéric Jurie. Modeling spatial layout with fisher vectors for image categorization. In *ICCV*, 2011.

[44] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011.

[45] Ivan Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005.

[46] Ivan Laptev, Marcin Marszałek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[47] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[48] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[49] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

[50] Subhransu Maji, Alexander C Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, 2008.

[51] Ionuţ Mironică, Ionuţ Duţă, Bogdan Ionescu, and Nicu Sebe. Beyond bag-of-words: Fast video classification with fisher kernel vector of locally aggregated descriptors. In *Multimedia and Expo (ICME)*, 2015.

[52] Ionuţ Mironică, Ionuţ Cosmin Duţă, Bogdan Ionescu, and Nicu Sebe. A modified vector of locally aggregated descriptors approach for fast video classification. *Multimedia Tools and Applications*, pages 1–28, 2016.

[53] Frank Moosmann, Eric Nowak, and Frederic Jurie. Randomized clustering forests for image classification. *TPAMI*, 30(9):1632–1646, 2008.

[54] Sanath Narayan and Kalpathi R Ramakrishnan. A cause and effect analysis of motion trajectories for modeling actions. In *CVPR*, 2014.

[55] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013.

[56] Eunbyung Park, Xufeng Han, Tamara L Berg, and Alexander C Berg. Combining multiple sources of knowledge in deep cnns for action recognition. In *WACV*, 2016.

[57] Xiaojiang Peng, Limin Wang, Yu Qiao, and Qiang Peng. Boosting vlad with supervised dictionary learning and high-order statistics. In *ECCV*. 2014.

[58] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *arXiv:1405.4506*, 2014.

[59] Xiaojiang Peng, Limin Wang, Xingxing Wang, and Yu Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *CVIU*, 150:109–125, 2016.

[60] Xiaojiang Peng, Changqing Zou, Yu Qiao, and Qiang Peng. Action recognition with stacked fisher vectors. In *ECCV*, 2014.

[61] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*. 2010.

[62] Stergios Poularakis, Konstantinos Avgerinakis, Alexia Briassouli, and Ioannis Kompatsiaris. Computationally efficient recognition of activities of daily living. In *ICIP*, 2015.

[63] Kishore K Reddy and Mubarak Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981, 2013.

[64] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013.

[65] Enver Sangineto. Pose and expression independent facial landmark localization using dense-surf and the hausdorff distance. *TPAMI*, 35(3):624–638, 2013.

[66] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *ICPR*, 2004.

[67] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM MM*, 2007.

[68] Jeong-Jik Seo, Hyung-Il Kim, Wesley De Neve, and Yong Man Ro. Effective and efficient human action recognition using dynamic frame skipping and trajectory rejection. *IVC*, 2016.

[69] Feng Shi, Emil Petriu, and Robert Laganiere. Sampling strategies for real-time action recognition. In *CVPR*, 2013.

[70] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[71] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[72] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth International Conference on*, pages 1470–1477, 2003.

[73] Alan F Smeaton, Paul Over, and Wessel Kraaij. Evaluation campaigns and trecvid. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, 2006.

[74] Cees GM Snoek, Marcel Worring, Jan C Van Gemert, Jan-Mark Geusebroek, and Arnold WM Smeulders. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *ACM MM*, 2006.

[75] Berkan Solmaz, Shayan Modiri Assari, and Mubarak Shah. Classifying web videos using a global video descriptor. *Machine vision and applications*, 2013.

[76] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[77] Deqing Sun, Stefan Roth, and Michael J Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *IJCV*, 106(2):115–137, 2014.

[78] Ju Sun, Xiao Wu, Shuicheng Yan, Loong-Fah Cheong, Tat-Seng Chua, and Jintao Li. Hierarchical spatio-temporal context modeling for action recognition. In *CVPR*, 2009.

[79] Lin Sun, Kui Jia, Dit-Yan Yeung, and Bertram E Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015.

[80] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[81] Jasper R. R. Uijlings, Ionut C. Duta, Negar Rostamzadeh, and Nicu Sebe. Realtime video classification using dense hof/hog. In *ICMR*, 2014.

[82] Jasper R. R. Uijlings, Ionut C. Duta, E Sangineto, and Nicu Sebe. Video classification with densely extracted hog/hof/mbh features: an evaluation of the accuracy/computational efficiency trade-off. *International Journal of Multimedia Information Retrieval*, 4(1):33–44, 2015.

[83] Jasper R. R. Uijlings, Arnold WM Smeulders, and Remko JH Scha. Real-time visual concept classification. *Transactions on Multimedia*, 12(7):665–681, 2010.

[84] Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *ACM Multimedia*, 2010.

[85] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[86] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79, 2013.

[87] Heng Wang, Dan Oneata, Jakob Verbeek, and Cordelia Schmid. A robust and efficient video representation for action recognition. *IJCV*, 2015.

[88] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[89] Heng Wang and Cordelia Schmid. Lear-inria submission for the thumos workshop. In *ICCV Workshop*, 2013.

[90] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.

[91] Limin Wang, Yu Qiao, and Xiaoou Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015.

[92] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015.

[93] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: towards good practices for deep action recognition. In *ECCV*, 2016.

[94] Zhongwen Xu, Yi Yang, and Alex G Hauptmann. A discriminative cnn video representation for event detection. In *CVPR*, 2015.

[95] Xiaodong Yang, Pavlo Molchanov, and Jan Kautz. Multilayer and multimodal fusion of deep neural networks for video classification. In *ACMMM*, 2016.

[96] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[97] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition*. 2007.

[98] Bowen Zhang, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang. Real-time action recognition with enhanced motion vector cnns. In *CVPR*, 2016.

[99] Xi Zhou, Kai Yu, Tong Zhang, and Thomas S Huang. Image classification using super-vector coding of local image descriptors. In *Computer Vision–ECCV 2010*, pages 141–154. 2010.

[100] Jun Zhu, Baoyuan Wang, Xiaokang Yang, Wenjun Zhang, and Zhuowen Tu. Action recognition with actons. In *ICCV*, 2013.